ONEedge.io

A Software-defined Edge Computing Solution

D2.1. Solution Framework - a

Solution Framework Report v.1.0

31 January 2020

Abstract

ONEedge is an open source platform for extending private cloud orchestration capabilities to resources at the edge. It is built upon OpenNebula and applies a distributed cloud model to dynamically, and on-demand, build and manage private edge clouds to run edge applications. This document describes the use cases and user requirements that are guiding the innovative development of ONEedge. It defines the main components of this edge computing platform, identifies the main software requirements derived from user requirements, and explains the test cases, methods and demonstration scenarios that are being employed for the verification of the new edge cloud functionalities.

This project has received funding from the European Union's Horizon 2020 Research and Innovation Programme under Grant Agreement No 880412.

Deliverable Metadata

Project Title:	A Software-defined Edge Computing Solution
Project Acronym:	ONEedge
Call:	H2020-SMEInst-2018-2020-2
Grant Agreement:	880412
WP number and Title:	WP2. User Success Management
Nature:	R: Report
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	31/1/2020
Actual Date of Delivery:	31/1/2020
Lead Authors:	Alberto P. Martí and Michael Abdou
Authors:	Sergio Betanzos, Ricardo Díaz, Vlastimil Holer, Ignacio M. Llorente, Jorge M. Lobo, Ángel L. Moya, Rubén S. Montero and Constantino Vázquez
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
1.0	31/1/2020	Submitted	First final version of the report

Peer Review History

Version	Peer Review Date	Reviewed By

[1] A deliverable can be in one of these stages: Draft, Peer-Reviewed, Submitted and Approved.

Summary of Changes from Previous Solution Framework Version

[First Version of Solution Framework]

Executive Summary

Document D2.1 is the first version of the Solution Framework Report in WP2 “User Success Management”. It includes the definition of validation cases and requirements (T2.1), framework and architecture (T2.2) and verification suite (T2.3) of the ONEedge platform.

This document describes the use cases that guide the ONEedge innovation project and identifies the main user requirements derived from these use cases. The main use cases considered are online gaming, the Internet of Things (IoT), video streaming, Artificial Intelligence processing and telecom Multi-access Edge Computing. We have also included a number of references to other sectors and environments in which the use of processing and storage capabilities at the edge will bring significant benefits and will improve current processes and the user experience.

The main use cases described in this document have been obtained from existing OpenNebula users and represent a set of archetypical circumstances and requirements that edge computing solutions should be able to address. The development of ONEedge will therefore be guided by the real needs that we have identified among those early adopters of edge computing solutions. This process guarantees that the evolution of the technological aspects of this project will always be driven by the requirements of its users.

The main principle that guides this development comes from the fact that latency is the key factor for quality of experience. In his document we detail the model for an application running at the edge, its basic components and characteristics, and we provide an overview of the distributed cloud edge infrastructure on which the applications are executed. This model implies the possibility of deploying the necessary resources at the edge plus the extension of private cloud orchestration capabilities in order to manage those resources effectively. ONEedge will be able to put in place a distributed cloud model and will provide companies and users with the necessary tools for, dynamically and on-demand, build and manage private edge clouds on which running their own edge applications.

This report also describes the main components that will be considered to build this edge computing platform, identifying the main software requirements derived from the user requirements that we have identified in early adopters of edge computing solutions:

The main components of the ONEedge architecture are:

- **EdgeScape** (Edge Instance Manager), in charge of installing, maintaining and monitoring the Edge Management Platform (EdgeNebula and EdgeProvision).
- **EdgeNebula** (Edge Workload Orchestration and Management), responsible for orchestrating the edge cloud infrastructure resources and managing the life-cycle of the application instances.
- **EdgeCatalog** (Edge Provider Selection), which maintains a list of edge resource providers that have been certified to work with ONEedge.
- **EdgeProvision** (Edge Infrastructure Allocation and Deployment), which allows a user to manage the full life-cycle of resources placed in independent edge locations, from their provision and maintenance to the final process of unprovision.
- **EdgeMarket** (Edge Applications Marketplace), the component that aggregates all the sources of virtual machines and container images plus the needed metadata.

Finally, this document also includes the specification of the verification scenarios and the definition of the test suite for the validation of these scenarios.

This analysis represents the starting point for work packages WP3 and WP4, which will specify, design, develop, test and verify the particular components to be enhanced and productized in order to satisfy the original list of requirements that guide the development of this project.

This Deliverable has been released at the end of the start phase (M3), and will be updated with incremental releases by the end of each business and product innovation cycle (M9, M15, M21).

Table of Contents

Abbreviations and Acronyms 8

1. Introduction 10

2. General Description 11

- 2.1. Innovative Approach 12
- 2.2. Simple Approach 13
- 2.3. Product Perspective 13

PART I. Use Case Analysis 15

3. Use Cases 15

- 3.1. Online Gaming (UC1) 15
- 3.2. Internet of Things (UC2) 16
- 3.3. Video Streaming (UC3) 18
- 3.4. Artificial Intelligence Processing (UC4) 19
- 3.5. Telecom Multi-access Edge Computing (UC5) 21
- 3.6. Other Use Cases 22

4. Use Case Requirements 25

PART II. Architecture Definition 28

5. Edge Execution Model 28

- 5.1 Edge Cloud Deployment Phase 29
- 5.2 Edge Application Deployment Phase 30

6. Distributed Edge Cloud Infrastructure Design 32

- 6.1. Edge Application Model 32
- 6.2. Edge Distributed Infrastructure 33
- 6.3. Edge Distributed Location 34
- 6.4. OpenNebula Implementation 38

7. Edge Platform Architecture 39

- 7.1. Edge Instance Manager (EdgeScape) 40
- 7.2. Edge Workload Orchestration and Management (EdgeNebula) 44
- 7.3. Edge Provider Selection (EdgeCatalog) 46
- 7.4. Edge Infrastructure Provision and Deployment (EdgeProvision) 49
- 7.5. Edge Apps Marketplaces (EdgeMarket) 54

8. Software Requirements 58

- 8.1. Edge Instance Manager (CPNT1) 58
- 8.2. Edge Workload Orchestration and Management (CPNT2) 59
- 8.3. Edge Provider Selection (CPNT3) 61

8.4. Edge Infrastructure Provision and Deployment (CPNT4) 62
8.5. Edge Apps Marketplace (CPNT5) 63
9. User to Software Requirements Matching 65
PART III. Software Certification and Verification Plan 68
10. Software Certification 68
10.1. Infrastructure 68
10.2. Certification Process 70
10.3. Drivers Certification 70
10.4. Software Delivery and Validation 71
10.5. Sandboxed Environments 71
11. Software Verification 72
11.1. Verification Methodology 72
11.2. Verification Scenarios 72
11.3. Integration Scenarios 77
12. Conclusions and Next Steps 78

Abbreviations and Acronyms

ACL	Access Control List
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
AWS	Amazon Web Services
CD	Continuous Delivery (Deployment)
CI	Continuous Integration
CLI	Command Line Interface
CM	Cloud Manager
CMP	Cloud Management Platform
CMS	Configuration Management System
CSP	Cloud Service Provider
CQRS	Command Query Responsibility Segregation
DB	Database
DDD	Domain-Driven Design
DHCP	Dynamic Host Configuration Protocol
DNS	Domain Name System
FW	Firewall
GUI	Graphical User Interface
HA	High Availability
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IP	Internet Protocol
JVM	Java Virtual Machine
L2	Layer 2
L3	Layer 3
LAN	Local Area Network
NFV	Network Function Virtualization
NM	Network Manager
OCA	OpenNebula Cloud API
OCCI	Open Cloud Computing Interface
OS	Operating System
PaaS	Platform as a Service
QoS	Quality of Service
REST	Representational State Transfer

RPC	Remote Procedure Call
SaaS	Software as a Service
SDN	Software Defined Network
SFC	Service Function Chain
SLA	Service Level Agreement
SOAP	Simple Object Access protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
UDP	User Datagram Protocol
UX	User Experience
VDC	Virtual Data Center
VLAN	Virtual Local Area Network
VM	Virtual Machine
VNF	Virtualized Network Function
VPN	Virtual Private Network
VXLAN	Virtual Extensible LAN
XML	Extensible Markup Language

1. Introduction

The purpose of deliverable D2.1 is to define the ONEedge edge computing platform for each of the three planned product innovation iterations. To this end, D2.1 will be released at the beginning of each development cycle at M3, M9 and M15 with an incremental definition of use cases and requirements (T2.1), framework and architecture (T2.2), and verification suite (T2.3). A final version of the report will be released at the end of the last cycle at M21.

D2.1 is a living document that is composed of two introductory sections and 9 additional sections organised in three main blocks of content:

- **Part I** focuses on the definition of user requirements extracted from case studies. Section 3 describes the different use cases of the ONEedge project. Section 4 identifies and defines the main user requirements for each of the ONEedge case studies.
- **Part II** focuses on the definition of the software requirements of the ONEedge framework. Section 5 defines the execution model offered by the edge computing platform. Section 6 describes the architecture of the edge infrastructure configuration that will be deployed on-demand on the provisioned edge resources. Section 7 describes the architecture and building blocks of the private edge computing framework. Section 8 presents the functional gaps and requirements grouped into the main building components. Section 9 collects the system requirements and the matching between the requirements posed on the system and the requirements imposed by the usage of the platform. .
- **Part III** focuses on the software certification and verification plan. Section 10 defines the software certification process and Section 11 presents a list of methods for verifying the software requirements.

The deliverable ends with a conclusion section.

2. General Description

So many companies now, both long-established and newly emerging, from multiple industries like gaming, Internet of Things (IoT), social networking and telecommunications, are focusing their business strategies on being able to provide innovative services and capabilities with absolute immediacy for their customers. Latency is the key factor for quality of experience. Even a delay of milliseconds is perceptible if an application requires instantaneous interaction.

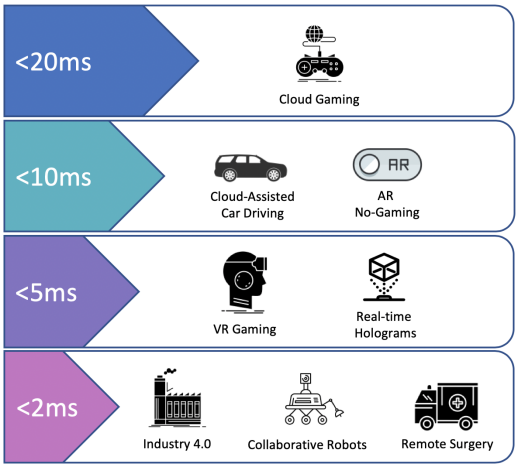

Figure 2.1. Estimated latency requirements in milliseconds for various types of applications.

"Edge computing" is the progressively maturing paradigm of shifting away from centralized processing towards carrying out the processing as close in proximity as possible to mobile devices, sensors, and end users. Yet, despite the urgent need to define a foundational platform that could deliver these edge services, and the dramatic growth in public interest and industry investment, at present, the companies that are developing next-generation solutions simply do not have a viable platform on which their low latency applications can become mainstream.

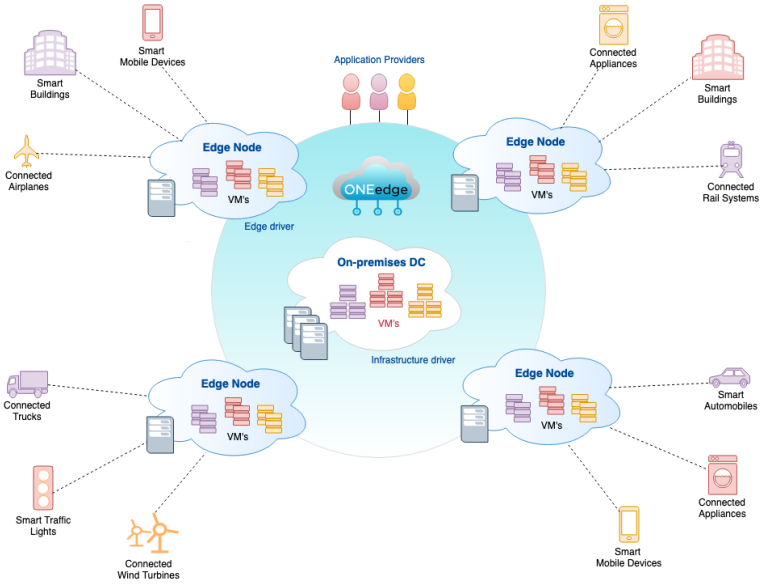

Figure 2.2. Architecture for disaggregated edge management.

ONEedge provides companies with an automated software-defined platform through which to build their own private, light and nimble edge computing environments based on highly-dispersed edge nodes in close proximity to users, machines, and sources of data. In this way, companies are able not only to easily create their own edge environments and manage them with utmost simplicity, but also to create these environments without having to provide or own those underlying resources at all.

2.1. Innovative Approach

ONEedge is an edge computing platform that builds and orchestrates a distributed edge cloud infrastructure to run private edge computing applications. Compared to other approaches and competing products, ONEedge is based on the following innovative methods and techniques:

- Where other solutions require an expensive and laborious process to stand up and configure an edge node, just as if it were another datacenter, ONEedge is completely **software-defined, programmatic and delivered 'as a service'** by using edge physical nodes from existing bare-metal cloud providers. All infrastructure components are represented in software form, including software-defined networking, virtual network functions, software-defined storage, and virtual machines or containers.
- Even better, the physical resources that comprise these edge cloud solutions do not need to be owned or provisioned by the edge cloud owner/administrator. ONEedge takes an **opportunistic approach to its edge node provisioning model** by taking advantage of already-available resources offered by third-party bare-metal providers.
- ONEedge offers a **Resource Provider Catalog** that maintains a list of edge resource providers which are certified to work with ONEedge. This catalog allows users to easily select which providers, locations and instances are better suited for their edge applications in terms of cost, capacity, latency, bandwidth, etc. On the other hand, providers are offered a centralized marketplace where they can promote the capabilities and costs of their certified locations and instances.
- **Multi-cloud interoperability** is achieved at the virtualized infrastructure layer. ONEedge is able to provide a uniform view of the underlying resources from different cloud providers. An application can be deployed anywhere on the edge infrastructure without performing any additional configuration or setup.
- While the main efforts of competing solutions has been focused on creating a single world-scale public cloud along the edge, ONEedge focuses instead on broadening **singular private clouds** and extending private cloud orchestration capabilities to resources at the edge. Its concentration on a **disaggregated cloud model**, which aims to include edge nodes as integrated components of a cloud, provides companies with the tools they need to easily manage small-to-medium edge infrastructures consisting of tens of thousands of resources.
- Simplicity of use is met by providing **access to existing Marketplaces with pre-built edge applications** such as PaaS environments, serverless platforms, container orchestration, etc.

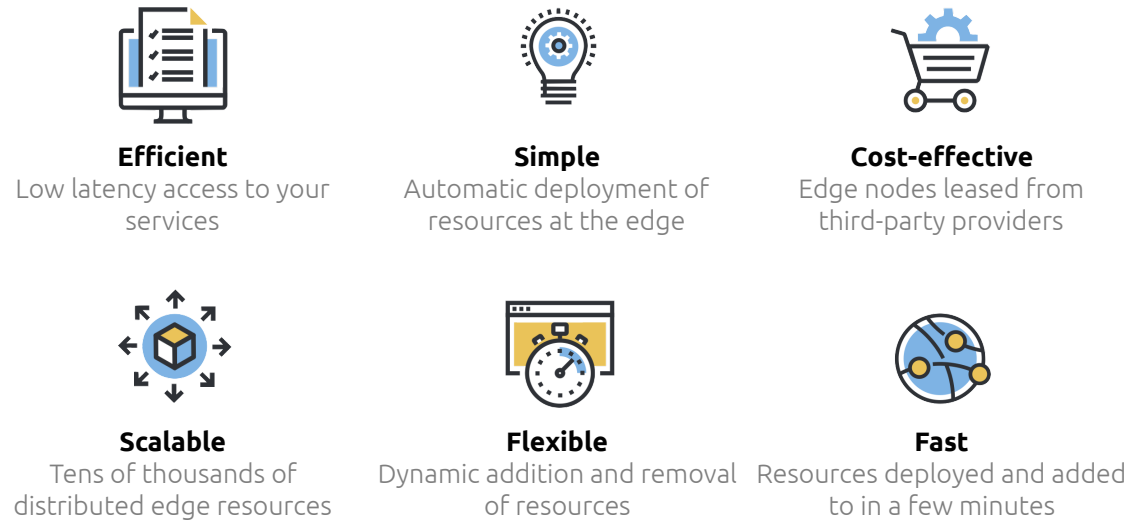

Figure 2.3. Benefits of the ONEedge approach.

2.2. Simple Approach

ONEedge brings simplicity to the edge infrastructure market by offering a simple, yet powerful solution to deploy edge environments built on both private, on-premises resources, as well as, edge resources offered on-demand by edge cloud providers.

ONEedge adopts the mix for a successful product: *low friction to try it out, a simple concept, focus, and useful functionality:*

- **Simple to Adopt.** Using edge resources on-demand from existing bare-metal cloud providers.
- **Simple to Install.** Management instance available as a Linux appliance or a managed service on a public cloud provider or on-premises.
- **Simple to Use.** Very simple graphical interface for consumers and administrators, simple provision model where remote edge resources are natural extensions of private resources, and integrated with existing VM and container marketplaces.
- **Simple to Update.** New versions can be easily updated with no downtime of the edge environments.

2.3. Product Perspective

ONEedge is being built as an extension of OpenNebula[2], a widely used open source cloud management platform.

- During the execution of the project, the ONEedge extensions will be **released as part of OpenNebula** and the support to these developments will be offered through the existing support channels offered by the open source project. The website ONEedge.io will offer specific documentation, white papers, use cases and datasheets. These materials will be designed to grow adoption and engagement, and to attract more users that could potentially, in return, contribute to validate the software.

[2] http://opennebula.io

- After project termination, ONEedge will be available as **Edgify**, a standalone software turnkey distribution for edge environments that will incorporate OpenNebula as its core cloud management platform. A new website Edgify.io will be created and separate promotional material will be produced so that we can address the target market with a differentiated, complete offer of product plus associated services.

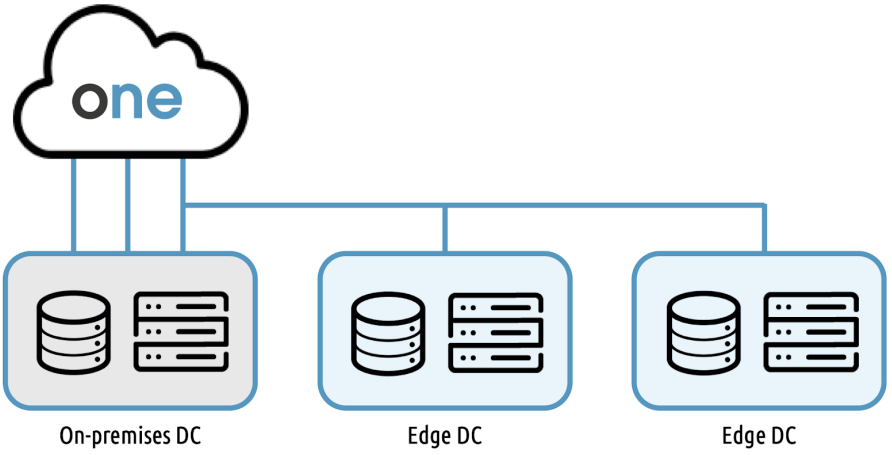

Figure 2.4. ONEedge extends private cloud orchestration capabilities to resources at the edge.

PART I. Use Case Analysis

3. Use Cases

These Use Cases have been defined from companies participating in the OpenNebula's Edge User Group. This User Group is being built with users of the OpenNebula cloud management platform that need a platform to offer application services at edge locations.

3.1. Online Gaming (UC1)

3.1.1. Background

The online gaming industry is reaping bountifully with an ever-growing audience and an impetus that is transforming the entertainment world. Video games have become mainstream, moving from the unwieldy consoles bought mainly for young kids to now being ubiquitous across mobile devices, and played by children and adults alike. The continual advancement of technology has brought a rapid growth in processing power, which has translated to more life-like graphics and animations, and in many cases, a more complete and engulfing gaming experience.

The video gaming population is undoubtedly expanding and evolving, investing more of their time in playing, as well as demanding increasingly more from their gaming experience. Not only are gamers looking for tantalizing graphics, but they are expecting a comprehensive, life-like experience, which oftentimes translates into "speed-of-thought" response times. Consequently, while online video gaming companies are developing more advanced games with increased data intensive processing, they are presented with the conflicting reality of distributing this huge amount of data to their players without any delay. Low latency has become one of the critical factors to the online video gaming experience.

3.1.2. Use Case Description

Several video gaming clients of ours have confirmed that this is their commercial reality. One key issue which they encounter comes at the launch of a new video game. They inevitably come up with a new game, and look to launch it to a global audience. However, especially at launch, they cannot be certain about how much demand will be generated and where it will be. So, with ONEedge, we have focused on providing these video gaming companies with the flexibility to automatically control the deployment of edge resources in accordance to where geographically they see actual, real-time demand.

Additionally, these video gaming companies are very clearly not interested in owning and managing the complete network of infrastructure hosting resources. Their main interest and commercial focus lies in developing a more dynamic video gaming experience for their users. And ONEedge provides the comprehensive set of capabilities allow these gaming companies to deploy their edge infrastructure by selecting the specific resource type, determining its hosted geographical location, and ultimately controlling the cost by leasing the resources and basing it all on their specific demand at any point in time.

3.1.3. Architectural Description

This gaming company requires the ability to create a global, singularly-managed distributed cloud infrastructure to host their video gaming servers, and to optimize user experience by providing a latency for their users below 10ms. They require the ability to deploy and configure resources at edge locations across the world and have them up and running within minutes, and to be able to manage the infrastructure with a flexibility to allow them to increase or decrease

resources at specific locations on-the-fly, which is tightly coupled to gaming demand at any point in time.

A single administrator should be able to create this edge cloud infrastructure, which may or may not include their own on-premises datacenter resources, and at the same time, take advantage of commercial bare-metal providers. Depending on actual user demand, they would have the flexibility to determine where geographically to deploy resources and configure is size and volume accordingly, all provided from a simple, singular user interface portal.

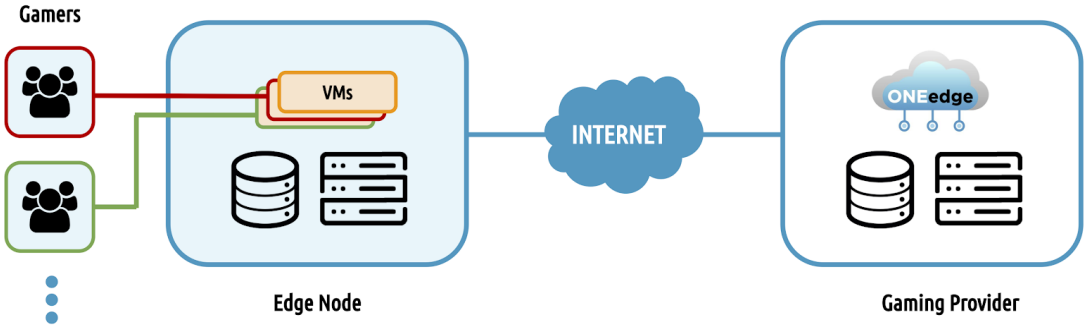

Figure 3.1. Online Video Gaming architecture.

3.2. Internet of Things (UC2)

3.2.1. Background

The inception of the Internet of Things (IoT) technology has amplified the cloud computing paradigm by establishing the means to be able to distribute computing and processing activities away from the centralized cloud. With the broad explosion of smart devices and their compute processing capabilities, technologies and their supporting organizations are focused on utilizing these distributed devices as more than just receptors of processed information, but rather making them active processors of information. This is critical to reducing latency and creating a much more “real-life” response time which is key to so many emerging technologies. Smart devices that are distributed across the globe cannot effectively depend on a centralized cloud to perform the cumulative analytical workload at the “speed of thought”.

Consequently, technologies like AWS IoT Greengrass provide the platform for edge nodes and devices, to act on data locally and to distribute analytical workloads and compute power across a broadly disseminated infrastructure, even with intermittent connectivity to a central cloud. Analytics and machine learning can effectively occur in close proximity to the devices generating data, distributing workload, and vastly reducing latency.

3.2.2. Use Case Description

We currently have a client whose primary business case is to provide a monitoring service of customers’ websites, utilizing AWS IoT Greengrass capabilities. Their services include being able to monitor and manage their customers’ site availability and performance metrics from across the globe. It is a solution that requires a largely distributed global infrastructure, allowing them to service their widely dispersed customers, and providing location-specific measurements and monitoring, as simple as calculating timings of website performance from various points of presence around the globe.

One key factor is that it is not in the interest of our client, nor is it financially feasible for them, to host their own edge resources across the globe, with measured geographical distance from every customer's hosted website, in order to be able to run analytical functions and execute monitoring operations. And while AWS gives a software stack the ability to connect edge nodes into the AWS IoT Greengrass framework, it doesn't offer the actual edge computing resources. The expectation is that the SDK and services are installed on the on-premises cloud within a company's datacenter or any preferred edge computing provider.

It is our ONEedge solution that will permit our client to create their own private distributed cloud in which they can easily deploy and manage edge nodes - on demand, and leased on usage - in geographical locations that are in close proximity to their customers' websites. When our client signs a new customer up for their monitoring service, they can determine the type of analytics that are needed, where they will need edge resources to best service the customer, and they can deploy and control configured edge nodes based on the current demand at those specific geographical locations. They have the ability to deploy AWS IoT Greengrass core services on those edge nodes, providing all of the functionality to run AWS Lambda functions, keep device data in sync, and efficiently carry out their global monitoring operations - all managed within their private cloud and without having to own the actual globally distributed edge resources.

3.2.2. Architectural Description

The model monitoring service of the Global Website Monitoring client of ours is divided into two parts. First are the "general services", which store the measured metrics, generate graphs and dashboards, alert the customers (e-mail, SMS), or trigger custom recovery actions. Second are the core distributed monitoring infrastructure with probes that are responsible for measuring how each customers' sites perform. While the general services can run in any single datacenter, the monitoring probes must run from as many different parts of the world as possible.

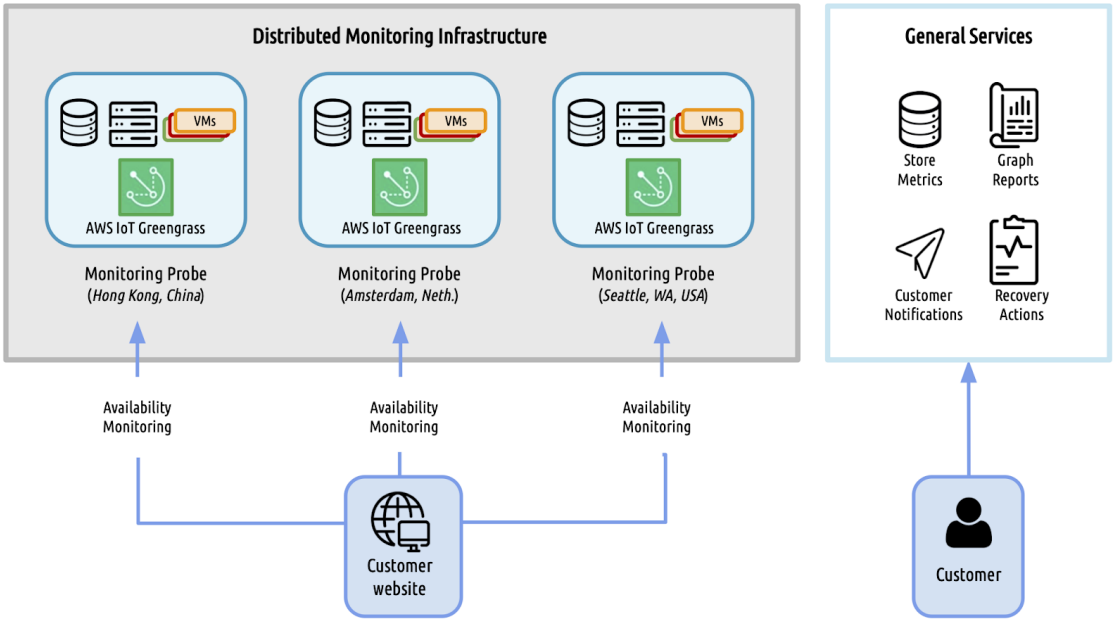

Figure 3.2. Sample Model Monitoring Service architecture.

ONEedge provides the ability to deploy custom edge nodes with the AWS IoT Greengrass core services, providing our client with an environment to control and run the monitoring probes, and take care of the transport of measurements to the high level processing services. It also provides the flexibility of utilizing resources of on-premises resources or to allow for the leasing of edge resources from bare-metal resource providers or public cloud providers.

3.3. Video Streaming (UC3)

3.3.1. Background

Over-the-top (OTT) media is revolutionizing the entertainment industry. We are seeing traditional HTTP streaming formats gain wide acceptance, facilitating the ability to reach viewers at a global scale on almost any device. Live events are being broadcast directly over the internet, allowing for immediate and widespread participation. However, there are technical limitations and challenges that accompany this new broadcasting method. For example, the demand for a live event can erupt quite quickly. Epic Games, the powerhouse video game creator of Fortnite, has established the regular practice of hosting global events for the gaming community to join and participate at scheduled times. The demand for these live events can jump from zero to millions of gamers within a matter of seconds.

The hyper-scale datacenters provided by traditional cloud service provide centralized computing resources. However, they are typically in a few locations, which are not usually in proximity to the densely populated areas where users reside. Video stream transfers from the centralized cloud to the dispersed users on a large scale can create huge bottlenecks in the startup of the streams, and translating into poor video quality, among other issues. Reliable and feasible edge computing solutions are in need to address the inherent scalability and reliability issues related to live streaming logistics.

3.3.2. Use Case Description

We have a client who has come up with the unique business case of media production at the edge. With the advent of 5G and an evolving media technology, they are looking to find ways to simplify the mechanics of television and other media production. Currently, being able to produce media for film or television that is not made in a singular studio location requires the deployment of a significant amount of technical equipment on-site, as well as a large production team.

Our client seeks to replace the need to displace a live TV truck and the production team by enabling their video equipment to connect directly to their edge computing environment. Firstly, with this solution, they would be able to dramatically reduce their production logistics and costs. At the same time, this proposed solution would enhance this media producer’s ability to broadcast live media to a globally distributed audience by providing a flexible edge hosting environment that could be managed and deployed in correlation to point-in-time demand.

3.3.3. Architectural Description

This broadcasting client requires a simplified architecture that will alleviate the need, first, to take their production equipment and team to on-site locations. Their camera equipment connects directly to their edge cloud nodes for media capture, while their technical production team can connect to the edge cloud from their production studio. Secondly, they are provided the flexibility to deploy edge nodes in close proximity both to locations of the broadcast event for edge connected production, as well as to locations of media demand. For video stream

broadcasting, edge resources can be strategically deployed based on the geographical location of actual demand for viewing.

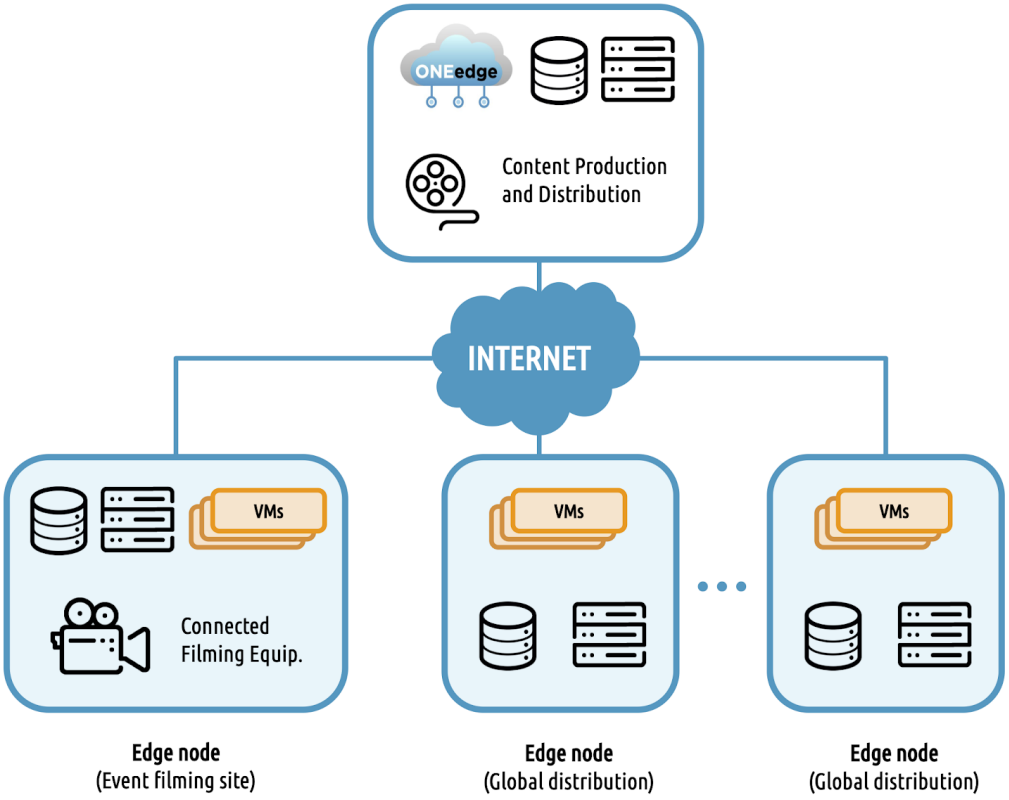

Figure 3.3. Video Streaming architecture.

3.4. Artificial Intelligence Processing (UC4)

3.4.1. Background

The ability for edge computing devices and systems to analyze data streaming in almost real time brings a significant improvement in the awareness and early response to the events that take place across a network. Now, with increasingly sophisticated processing capabilities moving to the edge, these distributed systems can analyze data for inference and pattern-detention much quicker. AI technologies—and Machine Learning in particular—can help to identify patterns and anomalies and can also make predictions on the basis of large datasets. This is proving to be particularly useful for the implementation of more effective cybersecurity measures.

The benefits of having AI-enhanced decision-making processes at the edge include:

- A response time closer to real time than the typical centralized cloud model, as data is processed most of the times within the same edge node.
- Clear separation between data generated at the edge and in the cloud, which keeps private information protected at the edge node and reduces the security vulnerabilities that arise from transmitting data to a central cloud.

- Support for the development of industry-specific or location-specific requirements, and compliance with local regulatory regimes (e.g. GDPR).

For AI techniques such as machine learning and deep learning models to deliver results speedily, they often rely on hardware accelerators such as graphics processing units (GPUs) or field programmable gate arrays (FPGAs).

3.4.2. Use Case Description

The rapid increase in the number of smartphones on the market and the widespread adoption of mobile technologies by the consumer makes the need for malware analysis on popular platforms like Android an urgent issue. Our user is a European technological start-up that is applying Artificial Intelligence (AI), Machine Learning (ML), data mining and deep learning techniques for malware analysis in Android platforms. Their solution includes anomaly detection and automatic deployment of security countermeasures (e.g. isolating the malware and alerting the users of a downloaded malware), avoiding the spreading of a detected malware to a larger community.

Our user's approach involves analyzing the behavior of Android applications, providing a framework to distinguish between applications that, even if they have the same name and version, behave in a different way. The aim is to detect applications behaving anomalously, thus detecting malware in the form, for instance, of trojan horses. The main contribution of this work is the implementation of a crowd-sourcing system to obtain the traces of applications' behavior. These anonymised and aggregated traces can then be used to separate benign applications from those trying to supplant the original ones and containing malware.

3.4.3. Architectural Description

Our user has developed a lightweight mobile client that can be downloaded and installed from Google Play Store. This application is in charge of monitoring Linux Kernel system calls and sending them preprocessed to a server. Users of this application contribute to the solution following a crowd-sourcing approach by sending (non-personal) behavior-related data of each application they use on their Android mobiles. These applications might have been obtained either through Google Play Store or maybe downloaded from a third-party unofficial repository.

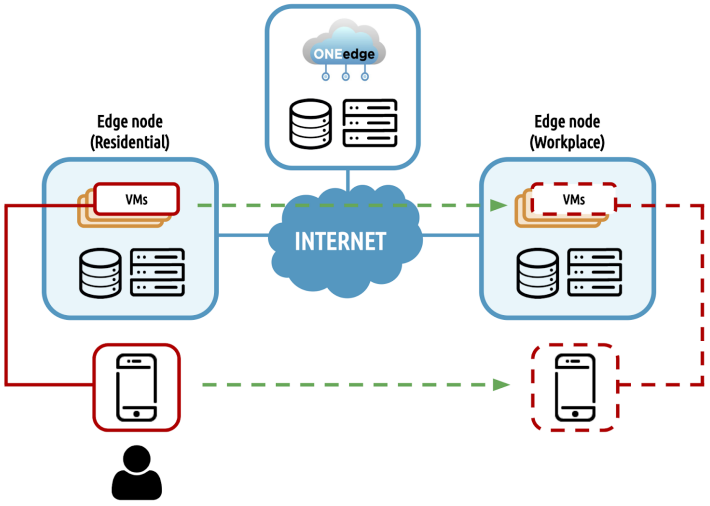

Figure 3.4. Artificial Intelligence Processing architecture.

Detecting security threats and being able as well to associate them with the locations of their origins are both important elements of the technological solution provided to its customers by our user. By being able to deploy their solution at the edge, our user can provide location awareness services and optime the response against potential threats, including the deployment of preventive measures to other Android platforms in the same region. Our user's solution combines static and dynamic analysis and applies Machine Learning techniques to assess the risk of unknown applications. Its edge-based machine learning framework is therefore crucial for the rapid detection of malicious applications. To accomplish that, our user takes advantage of the elasticity and flexibility of bare-metal edge providers in order to set up local deployments of their solution and assign edge computing and data storage resources dynamically.

3.5. Telecom Multi-access Edge Computing (UC5)

3.5.1. Description

Telco companies are starting to develop their own solutions to address the deficiencies that current centralized cloud services suffer. These include security (trust regulations often require critical applications and data to be served from the same country of the user), network cost (it is not economically viable to serve high volumes of content from a central location) and, above all, latency (the ultimate factor for quality of experience). Multi-Access Edge Computing (MEC) is a telecommunications networks architecture that involves the placement of cloud and IT resources in datacenters spread across a telecoms operator network. These datacenters can vary in size, location and capacity, and can be deployed within mobile, fixed, TV and/or enterprise networks.

This approach overcomes the traditional gap that, until now, has divided cloud and telecommunications networks. In the old model, cloud networks used to offer a set of IT tools and technologies to the end user, while telecommunications networks provided access to the services at the local level. With Multi-Access Edge Computing, telcos can make a new set of compute, storage and network capabilities available to customers at the edge of communications networks. In this way, services and applications are brought significantly closer to the end customers, improving the user experience and enabling new applications and offers.

3.5.2. Use Case Description

Our user is one of the largest telecommunication companies in Europe, who has been taking the first steps towards a Central Office Re-architected as a Datacenter (CORD) scenario.[3] CORD integrates network function virtualization (NFV) and software-defined networking (SDN) in order to reduce costs, bring cloud agility to the Telco Central Office, and refine control to the network. CORD's reference architecture is based on commodity hardware and a virtualization management platform to create and control the virtualized network functions.

Our user wants its Central Offices to be further re-architected as a cloud at the edge of the access network. As part of that process, the nature of their Central Office needs to evolve from a Connectivity Center to a Service Center. This will transform the Central Office into a multi-tenant environment where both our user and third-party providers can deploy applications and services with a great degree of control. The main goal is to bring the benefits of cloud computing and network programmability to the access network.

[3] https://opencord.org

This new architecture must support current residential services (e.g. Internet access, voice calls, and Internet Protocol Television) while allowing the deployment of third-party edge solutions. Our user also wants to use this process to redesign its strategy for managing Customer Premise Equipment (CPE) and for providing its fiber customers with self-provisioning services, abandoning the old model based on a physical installation of the CPE by a technician. For this, our user wants to adopt a solution based on open hardware, software agent integrated in the CPE, and a service logic hosted in the cloud. That provisioning process logic would run on the edge computing infrastructure that controls the access network, and would be capable of provisioning in real time the capacity and services that the fiber customer selects.

3.5.3. Architectural Description

ONEedge would be responsible for managing the virtualized resources that implement the different NFVs and edge applications. It would also interact with the rest of components of the Central Offices datacenter to establish the network connectivity for the VM on which the different services are running. Additionally, it would provide the orchestration functionality needed to manage multiple-VM applications that might include interdependencies and elasticity rules to dynamically adjust the number of VMs depending on the application load. Finally, ONEedge would also manage those virtualized components of the architecture and third-party edge applications that, for whatever reason (e.g. secure isolation) have to be deployed on independent VMs and/or in a separate network.

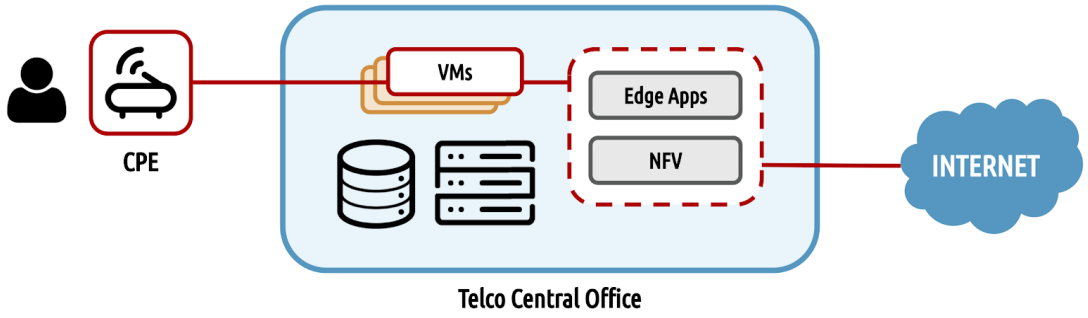

Figure 3.5. Multi-access Edge Computing architecture.

3.6. Other Use Cases

3.6.1. Virtual and Augmented Reality

One of the main advantages that edge computing brings for VR and AR experiences is the ability to reduce the poor user experience associated with low latency and slow frame refresh rates. AR services, for instance, need an application to be able to analyze in real time the output from a device's camera, combine that information with that associated with a specific location, and then produce the view that the end user will obtain when visiting that specific point of interest. The application needs awareness of a user's position and the direction they are looking in, provided via the device's camera and other internal sensors. Following a rapid analysis, the application must be able at that point to offer all the supplementary information in real time to the user. As soon as the user makes a movement, that new information needs to be updated. Hosting an AR service on an edge computing platform instead of in centralized cloud is beneficial because supplementary information relevant to a specific point of interest is

highly localized and quite often irrelevant beyond that particular area. The processing of all this information at the edge will benefit from a lower latency and a higher rate of data processing, thus improving the user experience.

3.6.2. Autonomous Vehicles

The response of an autonomous vehicle to an unexpected obstacle (e.g. a pedestrian crossing the street) needs to be immediate. In practical terms, it is not feasible nor scalable to locate all the necessary decision-making processing power at the remote infrastructure of a centralized cloud. Given the amount of data that autonomous cars are said to generate, and their dependence on quick analyses and low response times, dealing with that volume of data will certainly require edge computing. In this way, much of the workload will be done inside the car itself instead of relying on the existence of a permanent, stable link with a remote datacenter. Edge computing applied to autonomous vehicles opens the door not only to reduce latency by collecting and processing data locally, but also to establish an automatic interchange of information by distributing and sharing data in real time with other vehicles and devices in the vicinity.

3.6.3. Healthcare

Patients with chronic conditions are becoming increasingly dependent on wearable healthcare devices that provide real-time monitoring, early warnings and several methods for caregivers to be immediately alerted when help is required. Robotics is also playing an growing role in surgery, with devices that must be able to quickly analyze data in order to assist safely and accurately. End results could be fatal if these critical devices relied for a decision on sending data to a centralized cloud, especially if large volumes of information are involved. These devices require low-latency processing capabilities. On the other hand, there is enormous pressure to make sure that sensitive patient data is kept secure. Edge computing could be an alternative in those regulatory environments in which the use of a public cloud solution is not viable. Edge computing keeps data safe and secure on-premises, but incorporates the technology and benefits of the cloud.

3.6.4. Manufacturing Industry

With the digitalization of the manufacturing sector came an increased use of automation, software and data analytics, and vast amounts of data. IT (Information Technology) and OT (Operational Technology) convergence has now become a crucial part of this new scenario. Edge computing facilitates platforms for this convergence, enabling manufacturers to move away from closed, proprietary solutions. Manufacturers will be able to take full advantage of industrial IoT devices and put in place new applications that will be in charge of modifying production processes on the basis of real-time information, thus adapting more quickly to the current environment and reacting almost instantly to unexpected situations. The promise of predictive maintenance will be finally at hand, with manufacturers being able not only to collect huge amounts of data from their devices but, more importantly, to extract meaningful insights and integrate these automatically into their workflows (e.g. to schedule a maintenance visit to a specific machine). Edge computing, combined with 5G and new wireless technologies, will open up new possibilities to create smart factories, with connected robots and vehicles within the production facilities being operated remotely and internal processes being redesigned to increase productivity.

3.6.5. Construction

The application of edge computing in the construction industry will probably revolve around monitoring the development of new projects, ensuring worker safety and increasing

productivity. With the use of low-latency solutions, machinery in hazardous environments could be operated remotely, thus avoiding endangering construction workers. Drones and other devices can also be used to monitor the progress of a construction site, mitigate risks, analyze the environment and create real-time alerts, and ensure the project stays on track. All these applications depend on a rapid processing of large volumes of data collected on site, and cannot tolerate a delay in the transmission of that data or in its processing at a centralized cloud. Edge computing will provide those critical applications with the low latency and reliability that they require to operate.

4. Use Case Requirements

This section summarizes the user requirements from the use cases described in Section 3.

Id	Description	Source
UR0.1	Ability for users to discover and select among different edge locations comparing cost and performance.	All
UR0.2	Allow users to allocate edge resources from third-party bare-metal providers without having to own them outright.	All
UR0.3	Allow providers to easily offer their edge locations to users.	All
UR0.4	Ability to deploy resources globally at edge locations in close proximity to users.	All
UR0.5	Flexibility for users to be able to increase or decrease the deployed resources with ease at given specific edge location.	All
UR0.6	Allow users to determine the physical characteristics of the edge resources.	All
UR0.7	Allow users to determine the cost of the edge resources.	All
UR0.8	Users can import multi-components applications in a ready-to-deploy state.	All
UR0.9	Users can deploy application containers.	All
UR0.10	Users can easily upgrade a distributed edge infrastructure.	All
UR0.11	Users can easily install, configure and manage the edge platform stack.	All
UR0.12	Users can register to a customer service to get support, assistance and added-value to their edge infrastructure.	All
UR0.13	Edge nodes need to run with little or no intervention, so low (automatic) maintenance is essential.	All
UR0.14	Provide reliable and secure connectivity with edge resources.	All
UR0.15	Provide a simple graphical web user interface to perform management and operation tasks with ease.	All

Table 4.1. Common user requirements.

Id	Description	Source
UR1.1	Ability to define NUMA pinning for optimal performance.	UC1
UR1.2	Streamlined way to update the appliances, as video games have a high frequency update rate.	UC1
UR1.3	Ability to deploy applications (video games) with GPU passthrough.	UC1

Table 4.2. User requirements for UC1 (Online Gaming).

Id	Description	Source
UR2.1	Ability to deploy custom edge nodes with serverless platforms and micro-VM hypervisors.	UC2
UR2.2	Ability to combine edge resources with centralized cloud resources.	UC2
UR2.3	Integrate with pre-built IoT appliances for ease-of-use.	UC2

Table 4.3. User requirements for UC2 (Internet of Things).

Id	Description	Source
UR3.1	Provide the flexibility to deploy edge resources in close proximity to video editors and consumers.	UC3
UR3.2	Allow for camera equipment to connect to edge resources.	UC3
UR3.3	System should provide an interface for remote media content production and consumption.	UC3
UR3.4	Deploy appliances to be able to deliver synchronized interactive live online streaming, the RTMP feed must be converted to WebRTC on edge resources.	UC3

Table 4.4. User requirements for UC3 (Video Streaming).

Id	Description	Source
UR4.1	Need to execute lightweight execution environments (ideally micro-VMs) to analyze data provided by a mobile app.	UC4
UR4.2	Need to execute AI-based analyses at the edge.	UC4
UR4.3	Able to migrate end users' application data to different edge locations.	UC4
UR4.4	Able to respect local regulatory requirements for privacy and data protection.	UC4

Table 4.5. User requirements for UC4 (Artificial Intelligence Processing).

Id	Description	Source
UR5.1	Ability to define NUMA pinning for improved NFV-related performance.	UC5
UR5.2	Maximize NFV performance through the use of OVS and DPDK (user space traffic analysis and filtering).	UC5
UR5.3	Able to provide a high-density collation of NFV per server (ideally through the use of system containers).	UC5
UR5.4	Able to monitor high-density nodes and perform dynamic auto-scaling tasks.	UC5
UR5.5	Dynamic allocation of virtual networks for complex multi-VM services.	UC5

Table 4.6. User requirements for UC5 (Telecom Multi-access Edge Computing).

PART II. Architecture Definition

5. Edge Execution Model

ONEedge deploys and provisions resources at the edge and offers a simplified, efficient way to execute virtualized workloads (VMs / system containers) with minimum input from the user. Hence, the final goal of the ONEedge platform is to enable the creation of a distributed edge cloud and the deployment and orchestration of applications on this distributed cloud infrastructure. The model introduced in this section describes the workflow that ONEedge will follow in order to meet the Application Administrator's requirement of being able to deliver the application at the edge to provide low latency to the end user.

The model defines two different phases:

- The first one refers to the actions needed to perform in order to create a distributed edge cloud on a desired location or set of locations. These actions are performed by the **Infrastructure Administrator** and the phase is labeled as **Edge Cloud Deployment phase** (Section 5.1).
- The second phase refers to the set of actions needed to deploy an application or service in the edge cloud defined in the previous phase. These actions are performed by the **Application Administrator** and this step is labeled as **Edge Application Deployment phase** (Section 5.2).

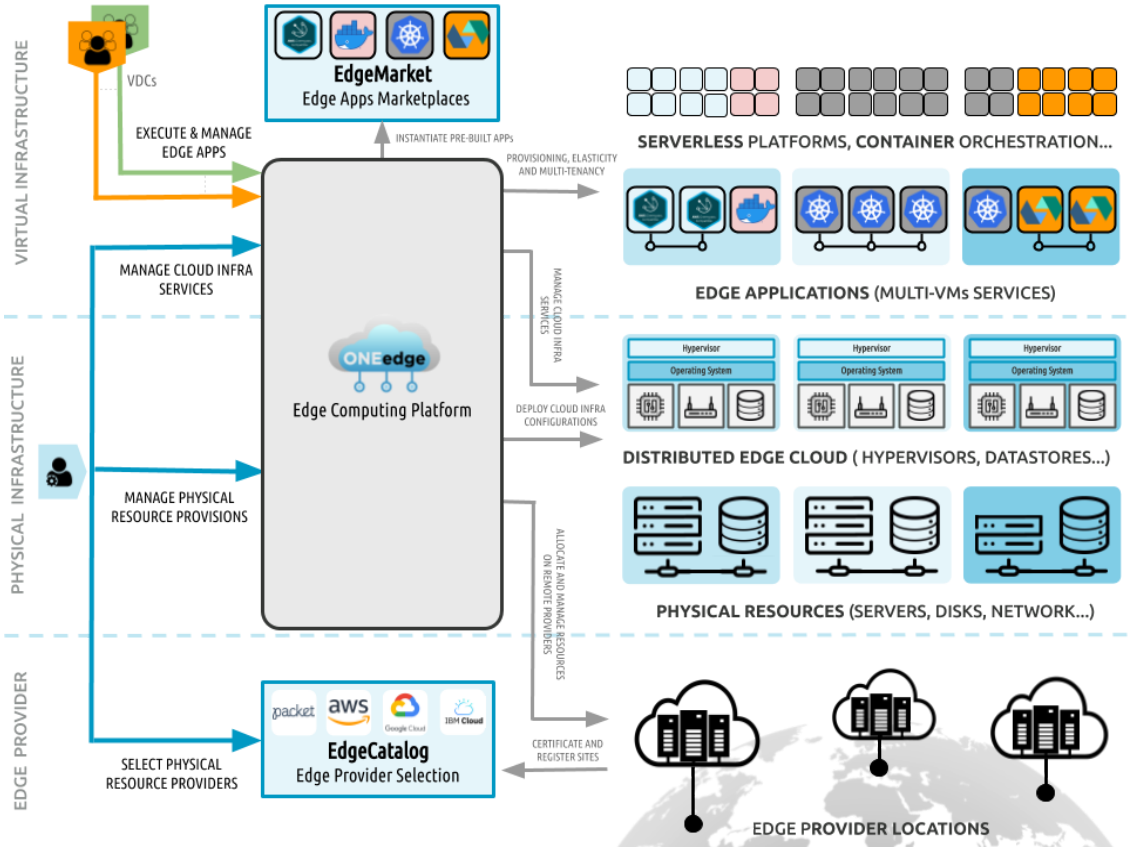

Figure 5.1. ONEedge general architecture view.

This model is designed to achieve the lowest possible friction between the different components involved. Each step hides the complexity of the underlying components and resources that are created, using pre-defined architectures for the Distributed Edge Cloud so the knowledge of the actual infrastructure details can be kept to a minimum.

5.1 Edge Cloud Deployment Phase

The starting point for building a distributed cloud infrastructure at the edge is the access to a ONEedge instance. This model assumes the simplest scenario where there is no private cloud on premises, but a ONEedge instance has been set up by the Infrastructure Administrator on a public cloud provider. Other scenarios are also possible, the most common being the existence of a private cloud on the company's premises. In this case, ONEedge would extend its native private cloud orchestration capabilities to those resources deployed at the edge.

The ONEedge instance with the core components needs to be deployed in advance by the Infrastructure Administrator. ONEedge core system is composed of several components that need to be deployed, namely the Edge Workload Orchestrator (EdgeNebula) and the Edge Resource Provision (EdgeProvision). As stated before, this system can be installed on premises or hosted on a cloud virtual machine (in AWS/EC2 for example). These components are automatically installed and maintained through the Edge Instance Manager (EdgeScape), which provides a simplified installation experience as well as automatic upgrades.

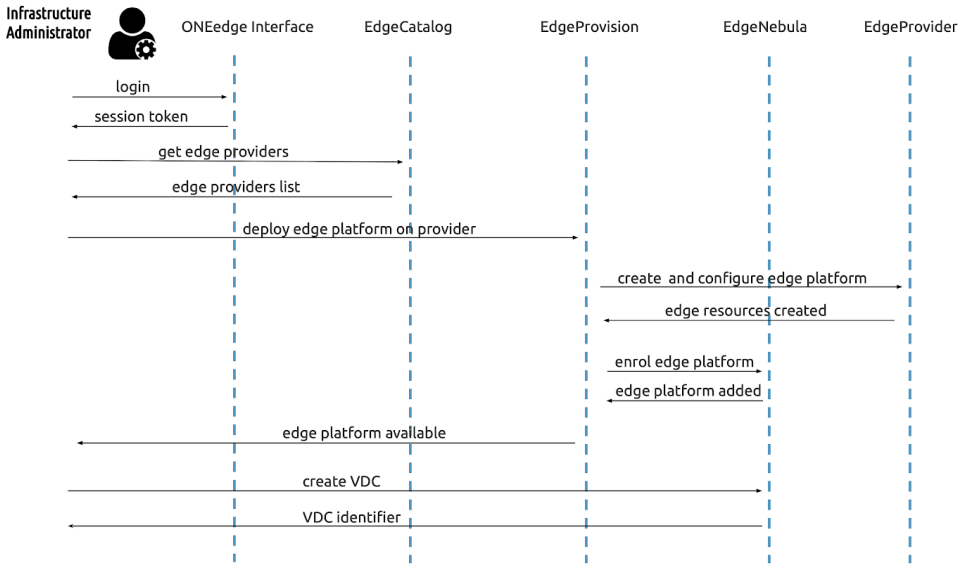

Figure 5.2. Infrastructure Administrator flow to create an edge cloud infrastructure with ONEedge.

The first step to deploy a distributed edge cloud is therefore logging into ONEedge using the Infrastructure Administrator credentials. The ONEedge web interface (easy to use modern web UI) presents the Infrastructure Administrator view, which follows a wizard-like approach. To select the best edge location, the interface will present a list of edge locations pulled from the EdgeCatalog component, along with information useful to make a decision: prices, availability, capacity, latencies, etc. This is coupled with tools and hints to aid and simplify the decision

making process to choose the best provider and location for the application that needs to be deployed in the edge.

Once an Edge Provider is (or Edge Providers are) selected, the EdgeProvision component offers an easy to consume functionality (through the ONEedge Interface or even the command line interface) to set up credentials of those providers and allocate the physical resources needed to build the desired Distributed Edge Cloud. Tools for the life cycle management of these edge cloud resources are also provided, as well as monitoring. Once the edge cloud infrastructure is created, its cloud platform services are enrolled in the main orchestrator component (EdgeNebula) in a transparent way.

The last step is used to create a Virtual Datacenter (VDC) on the EdgeNebula component that gives access to the new private edge cloud. With this VDC in place, the Application Administrator can start deploying the application on the edge.

5.2 Edge Application Deployment Phase

ONEedge defines two main types of Edge Applications, the single VM application (with access to a public IP) and the multi-VM application (with access to a public IP and optionally one or more private virtual networks). A multi-VM application represents a multi-tiered application, composed of interconnected Virtual Machines with deployment dependencies between them. The Edge Application (regardless of being single or multi VM) is deployed and managed as a single entity, and elasticity rules based on hypervisor and/or application metrics can be defined. Edge Applications can be backed by Virtual Machines or system containers. These Edge Applications can in turn be application container orchestration tools like Kubernetes, in which case the final application is deployed through this layer using for instance a Helm Chart.

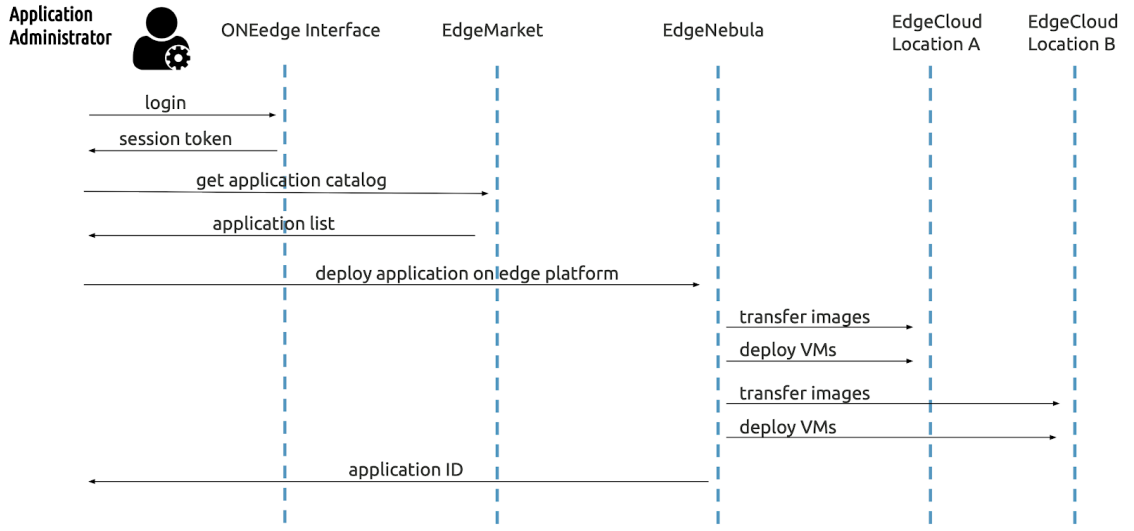

Figure 5.3. Application Administrator flow to create an edge cloud infrastructure with ONEedge.

After receiving the VDC identifier (and login credentials if needed) from the Infrastructure Administrator, the Application Administrator logs into the ONEedge interface to deploy the desired application on the Distributed Edge Cloud created in the previous phase. The Application Administrator is presented with a catalog of applications (pulled from the EdgeMarket component). She has then to decide which one of these pre-packaged applications

is going to be used to deploy the desired application. For instance, she can select a Kubernetes service to later deploy a Kubernetes pod on top, if the final application is for instance defined by a helm chart. ONEedge will then automatically download the Edge Application.

Once an application is selected, the EdgeNebula component is instructed to deploy the application on top of the Distributed Edge Cloud residing in a specific Edge Provider Location, selected by the Application Administrator. EdgeNebula then contacts this particular Distributed Edge Cloud location in order to:

- Transfer the images needed for the application
- Create the needed networks and address ranges within those Virtual Networks
- Create the needed Virtual Machines for the application (this can be 1 VM in the single VM case, or multiple VMs modelled as a OneFlow Template in the multi tier application case)
- Contextualize the Virtual Machines to configure the application

Once the application is deployed, EdgeNebula offers the functionality to control the lifecycle of the application as a whole.

6. Distributed Edge Cloud Infrastructure Design

In this Section we detail the model for an application running on the edge, its basic components and characteristics, Section 6.1. Then we provide an overview of the distributed edge infrastructure where the applications are executed, Section 6.2. The rest of this Section describes in detail the model assumed for each edge location.

The ultimate goal of the model presented here is to define an abstraction layer for the specific characteristics of edge providers. This model builds up a common ground to deploy edge applications in a simple way, as a natural extension of, and seamlessly integrated with, on-premises resources.

6.1. Edge Application Model

We assume that an instance of an edge application will only execute on one location. Note that multiple instances of the same application can be executed on multiple locations simultaneously

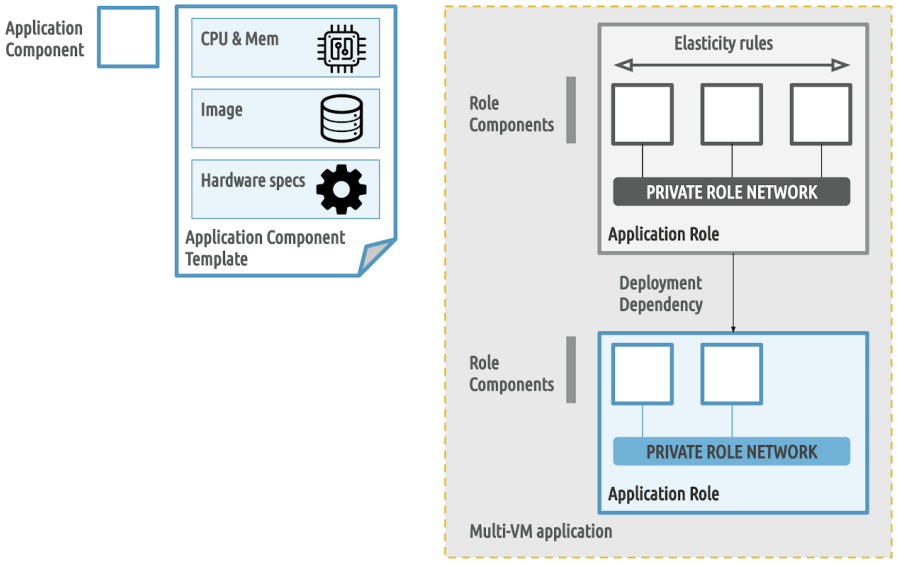

Figure 6.1. Main components of an edge application.

The simplest edge application consists of one single component, defined by:

- **Application image**: It contains the application runtime and the needed support for the operating system (OS). The OS pieces may depend on the compute node used by the edge location, see Section 6.3.
- **Public Internet** address.
- **Capacity specification** in terms of CPU and Memory.
- **Hardware requirements**: additional requirements to run the application, e.g. a specific processor layout or access to specific hardware like GPUs.

An application component is implemented by a single virtual machine (VM) or system container.

In general, an edge application will consist of more than one component as the one described above. In these multi-VM or multi-container applications, the application components are structured in tiers or roles. An application role is a set of application components (VMs) that implement a specific functional aspect of the application, and consist of:

- A **component template**, that includes the image, network connections, capacity specification and hardware requirements.
- **Deployment dependencies** for the role, e.g. roles that need to be running before deploying this role.
- Auto-scaling or **elasticity rules** that determine the number of components in the role over time.

Finally, apart from the roles, a multi-VM application may also include one **private network** to interconnect the components of the application. Figure 6.1 represents the concepts introduced in this section for a single application and a classical two-tier multi-VM application.

6.2. Edge Distributed Infrastructure

An edge application is executed on a distributed edge infrastructure that consists of a set of edge locations interconnected through a public Internet link, see Figure 6.2. This infrastructure may combine optionally a set of on-premises resources.

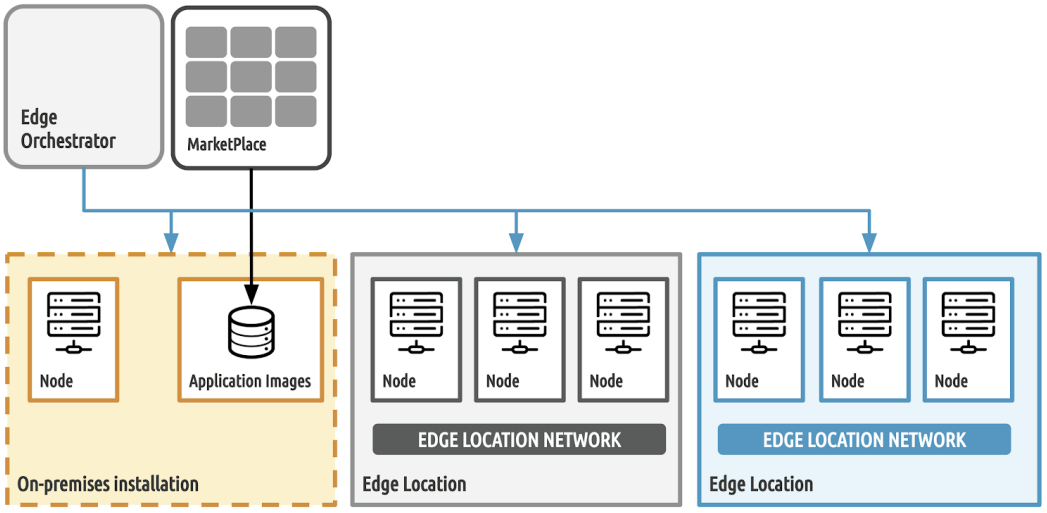

Figure 6.2. Main components of a distributed edge infrastructure.

The edge orchestrator is responsible for coordinating the use of the set of distributed locations, providing an uniform view of the underlying resources. In this way, an application can be deployed anywhere in the edge infrastructure without performing any additional configuration or setup. The following assumptions and requirements will be made in order to achieve this goal:

- Application images should be installed once and ready to be deployed anywhere.

- Applications will run from a single edge location. This restriction will be relaxed at a later stage of the project.
- On-premises resources are optional and not required to run the edge infrastructure.
- The edge orchestration platform only requires access to the edge locations and can be run anywhere within the edge infrastructure.
- A common storage service is used to store the application images. This storage area acts as an image repository for the edge applications, usually installed from the marketplace.

6.3. Edge Distributed Location

The basic building block of an edge cloud location is the **edge node** or server. In general, we assume that these edge nodes are bare-metal servers for exclusive use, provisioned by public providers through well-defined APIs. This allows the adoption of a dynamic and flexible model able to shape the edge location to the application needs. However, these providers also impose restrictions on the provisioned infrastructure. The limitations include, but are not restricted to:

- Operating system versions available for the edge nodes.
- Networking limitations for private traffic (e.g. between the edge nodes).
- Variable provision times.
- Different server offers in terms of the functionality available, e.g. hardware features (e.g. GPU support), or associated infrastructure services (e.g. NAS or distributed storage clusters).

Moreover, the functionality, API formats, options and limitations vary from provider to provider, which renders a very heterogeneous edge provider marketplace. These attributes will be included as part of the edge provider description in the Provider Catalog, see Section 7.3.1, to establish filtering and selection policies to deploy edge applications.

6.3.1. Compute Model

Each edge location is made of a set of bare-metal servers. Each bare-metal server is defined by the following characteristics:

- **Operating System** (OS) distribution and version, including the kernel version, basic user tools.
- **Compute specifications** including the amount of memory available in the server and number of CPU cores.
- **Storage and Network capacity** for the server local storage and network link performance
- **Additional hardware features** that may be optionally exposed to the application instances, e.g. PCI-passthrough or SR-IOV of GPU or network devices

On top of the bare-metal server and the basic OS installation (provisioned by the edge provider), a virtualization layer is required to deploy the applications. This layer comprises a suitable hypervisor technology as well as any other software needed to create the platform services to run application instances. In particular we will consider two different technologies:

- **Virtual Machines** (VM). Fully isolated application environments running on full virtualization hypervisors. VMs provide the most secure environment for multi-tenant applications running on the edge at the cost of some overhead. However, for certain

application domains (e.g. serverless, functions-as-a-services…) that still require strong isolation, the VMs can be simplified to reduce such overhead. In particular ONEedge will support two VM types:

  - **Regular VMs** with full control to define the characteristics of the virtual server (e.g. number of network interfaces, virtual NUMA nodes, etc…)
  - **Micro-VMs** with a restricted feature and hardware set which provides a light and fast provision application instances.

- **Infrastructure Containers**: compared to VMs they offer a less resource-consuming solution while providing a full operation at infrastructure level (e.g. network models). The performance gains come at the cost of a lower degree of isolation and OS flexibility, as they run in the same kernel space.

Figure 6.3 summarizes the main components of the edge node.

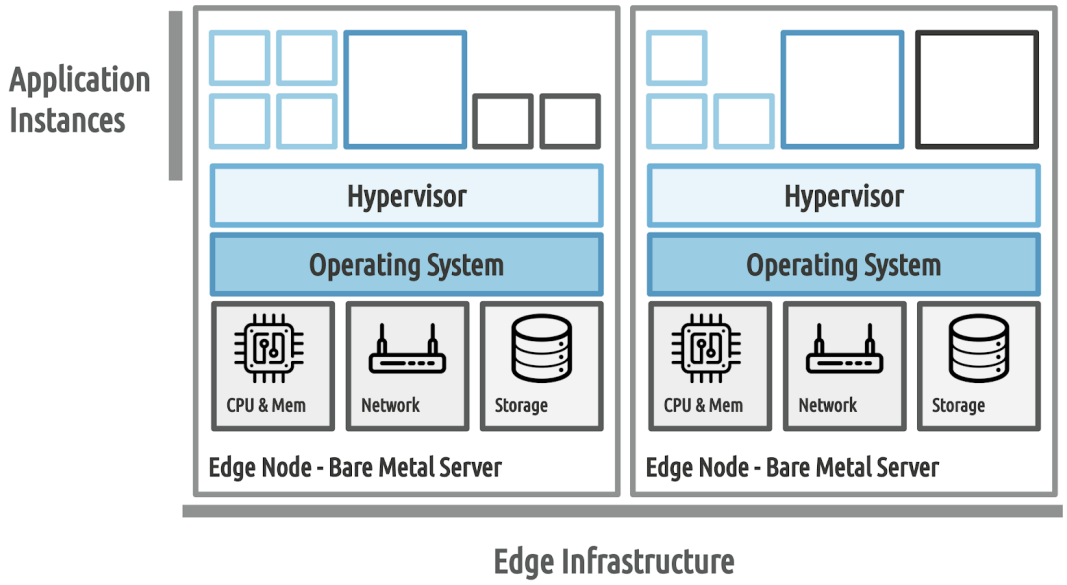

Figure 6.3. Main components of an edge node.

Table 6.1 describes the specific combination of software components and OS that will be used for edge node reference stack.

Edge Operating Systems	Hypervisor
CentOS	LXD
Ubuntu	KVM (qemu-libvirt)
ESXi	Firecracker
	vCenter VMware

Table 6.1. OS and hypervisors considered for the edge node.

6.3.2. Network Model

Edge providers offer a wide range of possibilities to interconnect each bare-metal instance, ranging from public networks to isolated VLAN networks that may optionally add a broad set of features to the network topology, like public gateways, VPN and DHCP servers or NAT gateways, among others.

However, these networking offerings do not consider in general the requirements to execute multiple VMs or containers within the node, as the providers usually impose some limitations to the traffic generated by the edge node. Moreover, the virtual networks required by the application instances are dynamic and usually require the definition of specific routes and the creation of network overlays.

We assume that each edge node is capable at least of connecting to the Public Internet, and route inbound/outbound traffic for multiple public IPs. Additionally the edge node may have one or more interfaces to interconnect to other edge nodes.

As a first approach we will adopt a simple model that consists of a public network shared across all application instances, and one or more private networks for private communication across application instance components. This models prioritizes:

- Networking performance for the private traffic across application instances.
- Simple deployment, operation and monitoring of private and public traffic.
- Portability across providers by only relying on basic networking features.

Figure 6.4 depicts the overall network topology considered here. The public network requires the definition of a public IP range in the the public provider so the IPs in that range can be routed through the provider backbone. Additionally the edge node needs to configure SNAT and DNAT to route the traffic internally to/from the application instance.

The private networks are defined internally by creating dedicated bridges to where the instances are attached, if required. Note that VMs in the same private network are co-allocated in the same edge node.

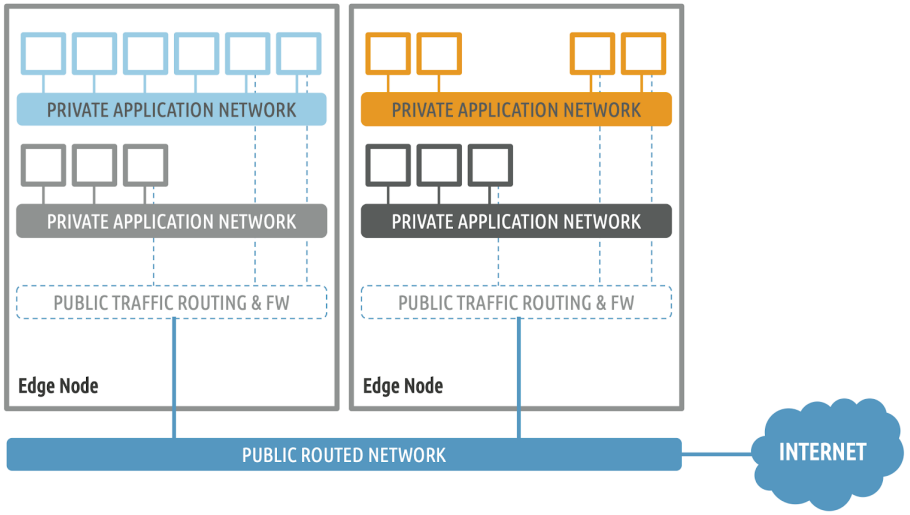

Figure 6.4. Network platform services and model of the edge infrastructure.

The next iteration of the infrastructure model will consider the implementation of private networks across edge nodes and locations.

6.3.3. Storage Model

The storage resources of the edge nodes are devoted mainly to store the application images that are running in each node. Given the specific nature of an edge infrastructure - dynamic and highly distributed - we assume that:

- Application instances are stateless and run from quasi-static base images.
- Regular operational procedures can be applied to the base images, e.g. backup, upgrade or security patch policies.

Base images for the applications are stored in an image datastore that serves as an image catalog for the applications. This image catalog is centralized and provides a uniform view of the images available to use in the edge infrastructure.

However, it would not be feasible to transfer the application images each time an application is started, as the transfer times will impose a high overhead both, in the deployment time, as well as in the overall cost of the edge infrastructure (because of the bandwidth consumption).

The edge infrastructure includes three different datastores:

- **Image datastore**, a repository of application images. Images are generally obtained from the Edge Apps Marketplace, see Section 7.5
- **Cache datastore**, used to store a copy of images in the Image datastore.
- **System datastore**, to store the images of the running application instances.

Note that the System and Cache datastores are local to the edge node and implemented using its storage resources. Transfers only occur between the Image and Cache datastores. Figure 6.5 outlines the Datastores and the relationship between them.

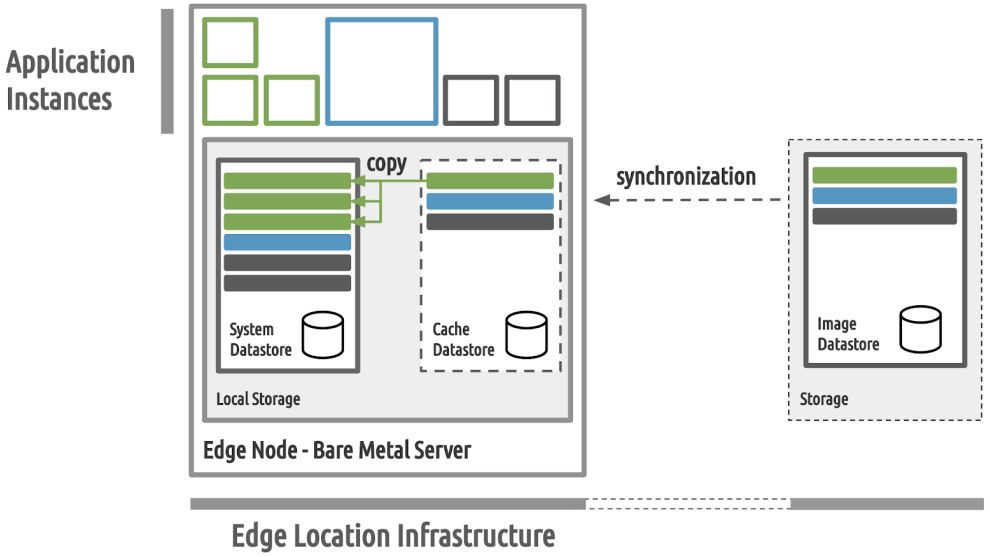

Figure 6.5. Edge infrastructure datastores and node storage.

6.4. OpenNebula Implementation

The Edge infrastructure defined above is implemented through a set of OpenNebula resources. In order to enforce allocation policies, (e.g. an application and its networks are all created in the desired node of a given location), the resources are structured in OpenNebula clusters. Each cluster will share some of the resources with other clusters within the edge location and across the edge infrastructure so applications can be easily deployed anywhere on the edge.

In particular, the edge infrastructure consists of (see Figure 6.6 for a graphical overview):

- **Image Datastore**, where application images are registered, typically, from a marketplace catalog. The Image Datastore is shared across locations and uses the drivers described in Section 6.3.3.
- **System Datastore**, used to run applications from the local storage area of the node. Images will be cached in its associated cache area.
- **Public Networks**, that contain public routable IPs for the applications. Public networks are shared across all the clusters in a given location, but not across them.
- **Hosts**, each edge node will be part of its own cluster, establishing a one-to-one relationship between clusters and hosts.
- **Private Networks**, created dynamically to support intra-application component communication. Private nets are restricted to the cluster where the app is deployed.
- **Clusters**, combine resources needed to run applications and enforce allocation policies. A cluster will therefore consist of: a host, private networks, public networks (shared with other clusters in the same location), image datastore (shared with all the clusters in the edge infrastructure), and a system datastore.

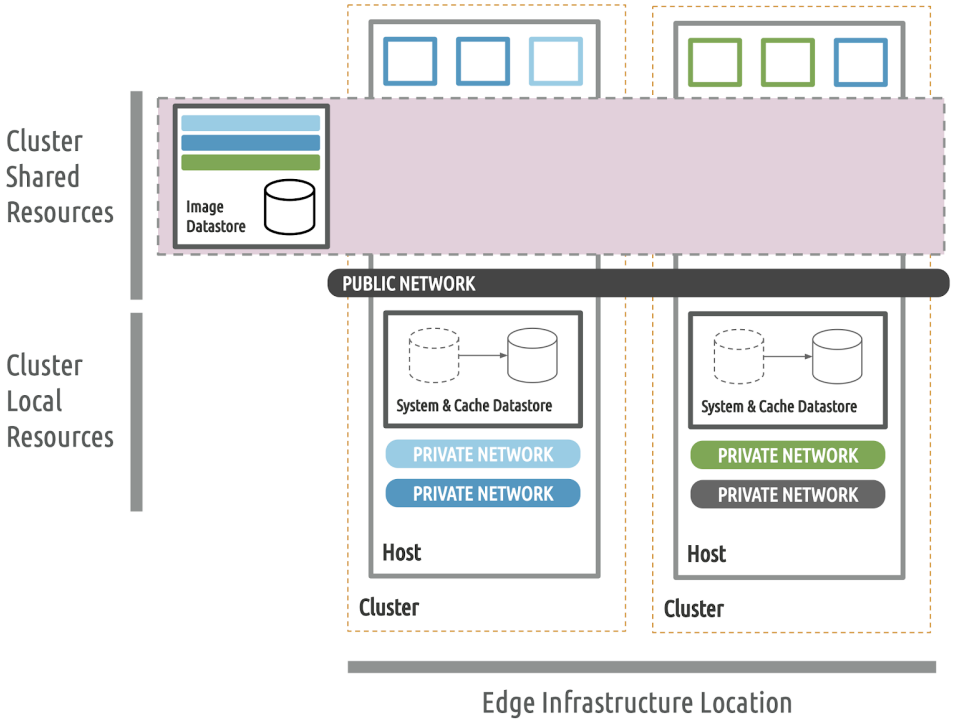

Figure 6.6. OpenNebula resources used to define the edge infrastructure

7. Edge Platform Architecture

This Section describes the main components of the ONEedge architecture.

- **EdgeScape** (Edge Instance Manager) installs, maintains and monitors the Edge Management Platform (EdgeNebula and EdgeProvision).
- **EdgeNebula** (Edge Workload Orchestration and Management) is responsible for orchestrating the edge cloud infrastructure resources and managing the life-cycle of the application instances.
- **EdgeCatalog** (Edge Provider Selection) maintains a list of edge resource providers which are certified to work with ONEedge.
- **EdgeProvision** (Edge Infrastructure Allocation and Deployment) allows to manage the full life-cycle of the complete independent edge locations, starting with their provision, maintenance until the unprovision.
- **EdgeMarket** (Edge Applications Marketplace) component aggregates all the sources of virtual machines and container images plus the needed metadata.

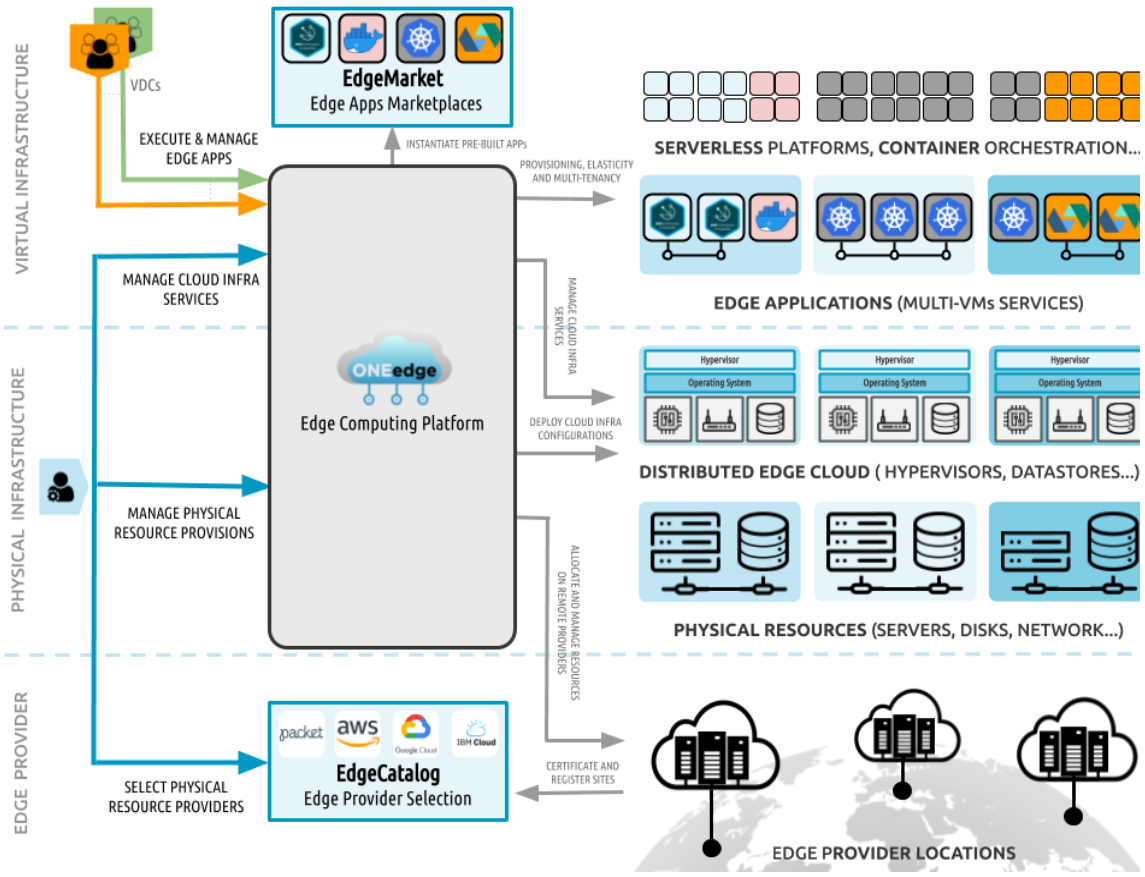

Figure 7.1. ONEedge general architecture view.

7.1. Edge Instance Manager (EdgeScape)

The **Edge Instance Manager** is a GUI based tool for the Infrastructure Administrators to deploy and manage the **Edge Management Platform** (Edge Workload Orchestration and Edge Resource Provision) along with various components. It accelerates the complex, time-consuming, error-prone task of deploying and managing the computing environment.

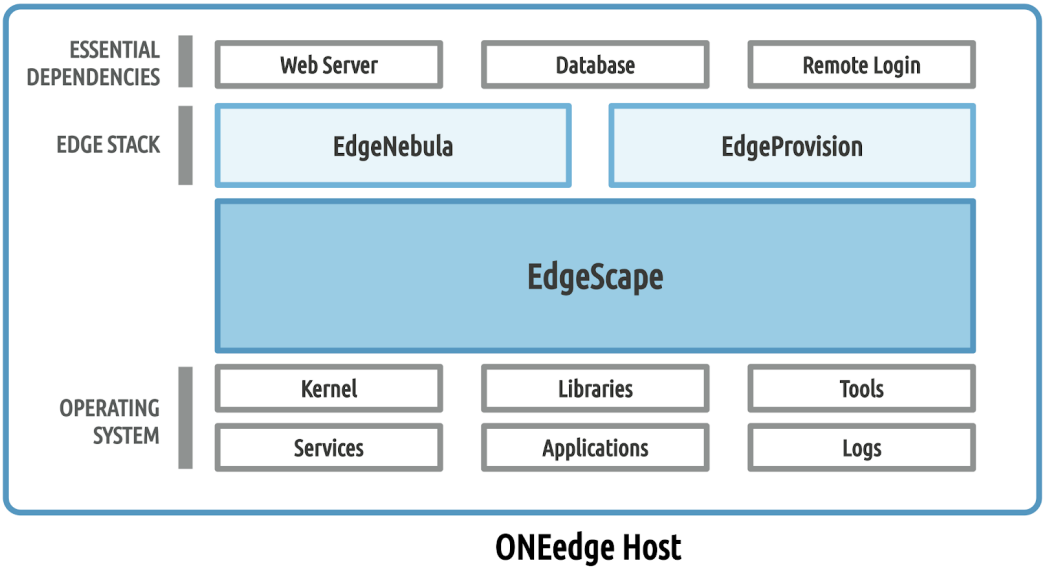

Figure 7.2. Collocated ONEedge host components.

It is a central management component, one of the first installed components on a custom dedicated host (physical or virtual machine, on-premises or cloud based). It provides the control interface for initial bootstrap (installation and configuration) of the edge stack (EdgeNebula and EdgeProvision), operating system and other dependencies. For existing installations, the same interface allows one to manage, reconfigure or update all components to the newer version. Integrated monitoring gives Infrastructure Administrators visibility into the health state of the system.

Main responsibilities of the Edge Instance Manager are:

- **Initial bootstrap** of edge stack components and their dependencies
- **Reconfiguration** of all involved parts following cloud administrator preferences
- **Upgrade** all parts of existing deployment
- **Monitor** core components, validate running installation, and provide access to logs
- **Subscription management** to link to the customer portal and paid subscription

7.1.1. Simplified Installation

Initial installation is divided into 2 steps by components:

1) **EdgeScape**
2) **Edge stack components** (EdgeNebula, EdgeProvision) and dependencies

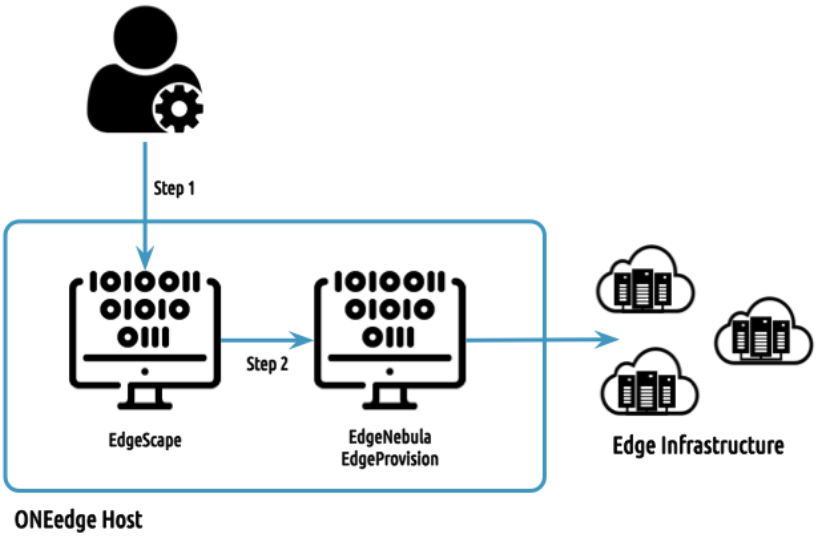

Figure 7.3. Installation Workflow.

Step 1: The installation of the EdgeScape itself is provided in multiple ways - the most straightforward is using the **operating system packages** and the package repository, providing access also to the future updates.

Step 2: The edge stack installation is managed internally by EdgeScape. The low-level technical installation details are hidden to the systems administrator behind a simple installation wizard summarizing main installation steps and progress. When installation is finished, the dedicated host is fully ready to use and resources are deployed on remote edge locations.

Significant installation tasks:

- Pre-installation validation
- Configure operating system
- Install and configure dependencies (database, web-server)
- Install and configure edge stack components (EdgeNebula, EdgeProvision)
- Start services
- Post-installation validation

The most suitable of current approaches is going to be used to handle the services bootstrap. The following points present the options which will be considered:

Software Delivery - ways to distribute components for installation

- **Operating System Packages** and repositories are the most straightforward means to provide the software and to allow future updates. They transparently integrate with the existing software and provide approaches users might have already installed on the system. Examples: rpm (yum), deb (apt).
- **Container Images** allow one to easily distribute and run software as a filesystem snapshot of the software with a limited minimal set of dependencies. Examples: Docker, OCI image formats.

- **Universal Operating System Packages** combine benefits of operating system packages (transparent integration into the operating system) and container images (self-contained distribution). Examples: snap, Flatpak.

Orchestration - installation and configuration

- **Shell Scripts** for automation of simple tasks are easy to write and maintain. Examples: bash, generic POSIX-like shells.
- **Configuration Management System** are tools which ensure the host complies with a defined state. They are used to manage the services installation, configuration and start. CMS will be used to manage all essential dependencies to run the edge stack. Examples: Ansible, Puppet, Chef.
- **Custom implementation** of individual operations related to edge stack installation is necessary inside the EdgeScape code. This provides the pre-installation step feedback on failure with an immediate and most-suitable recovery decision (retry or rollback).

Although the installation is divided into 2 steps based on different components involved, the final installation process can be simplified by providing a dedicated (interactive) **installation script** to hide the low-level details for the user. Such a script can manage just step 1 or both steps 1+2, which results in an unattended installation experience similar to what the tool **miniONE**[4] for single-node cloud evaluation deployments offers.

A final system snapshot with all components preinstalled can be provided, if necessary, as a **pre-built appliance** to import and run on popular virtualization platforms and hosting providers (VMware, Amazon EC2, or Google Compute Engine).

7.1.2. Automatic Upgrade

Single-click upgrades of all components is a crucial feature of the Edge Instance Manager to ensure the platform stays **current** and **secure** for all users.

Upgrades will be implemented as an extended installation process (7.1.1) with respect to the existing installation, extra validation steps and a **rollback** scenario, in case of failure. Here is a proposed workflow:

- Pre-upgrade validation
- Safe shutdown
- Backup
- Upgrade and reconfigure (local) edge stack components
- Upgrade remote edge locations
- Start services
- Post-upgrade validation

In case of upgrade failure, the edge stack will be restored to the starting point. Each state-changing action will provide a validation action and a reverse operation to undo the change on failure.

State of the upgrade process can be observed by the systems administrator from EdgeScape management interface, but users can't affect or modify the upgrade process.

[4] https://github.com/opennebula/minione

7.1.3. Instance Management

Except for the installation and upgrade of the whole edge computing environment, EdgeScape is going to be a central control point of the edge services running on a dedicated host. Therefore it must provide instance management features, e.g.:

- **Secured access** (authenticated and encrypted)
- **Reconfiguration** of operating system and edge services from a set of available options
- Access to **logs** and generate **debug bundle**
- Services health **monitoring**
- (Automatic) **Upgrade** for the operating system
- **Subscription management** to link the customer support services

For the operating system and edge stack **reconfiguration**, the same installation and configuration functionalities (points 7.1.1, 7.1.2) are going to be reused. New logic will include the (automatic) operating system upgrade.

Providing easy access to the system and edge stack logs is required for troubleshooting of the error situations. The control panel should moreover provide a mechanism to generate a debug bundle - a snapshot of the configurations, logs, and state of all related components on the EdgeScape instance and/or including the (selected) edge location(s). The OneSupport tool will be used as a base, and will be enhanced for use within the edge stack.

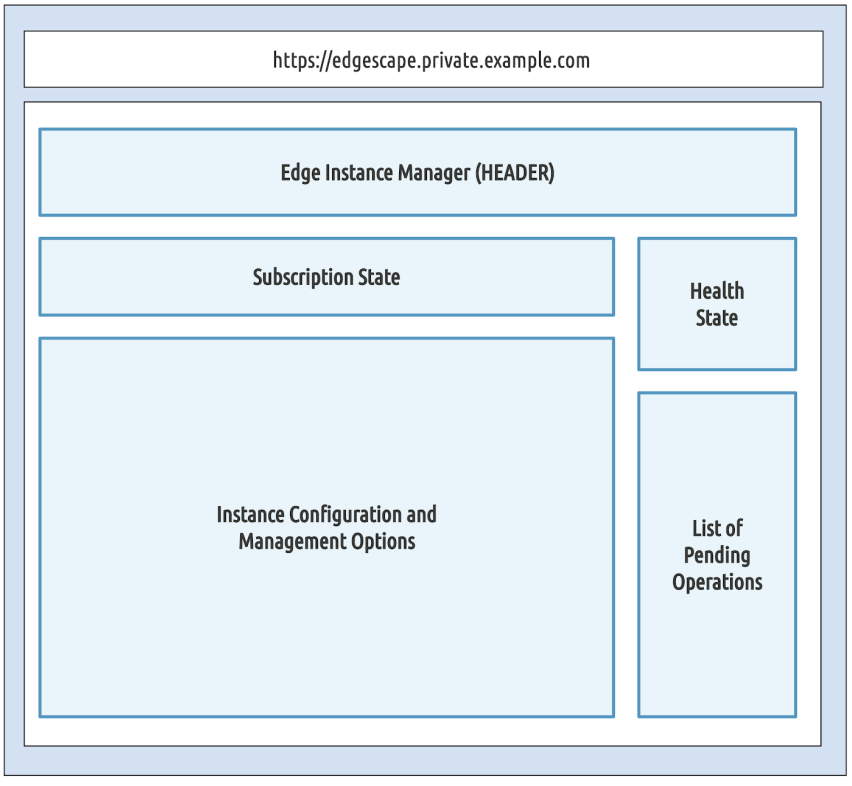

Figure 7.4. Wireframe of main EdgeScape control panel.

The control panel's integrated **monitoring** mechanisms provide visibility into the health state of individual services and aggregated state of the system. The health state can be observed on the web of control panel or via API to easily connect to the 3rd party monitoring systems.

Additional support services might be provided to the deployments which are properly linked with the customer accounts. Each installation must be uniquely identified to prevent abuse of the support subscriptions. To this effect, each ONEedge instance will generate a unique identifier (UUID) based on different parameters, including hardware identifiers. Out of the command line interface commands will be signed with this unique identifier to prevent giving support to other instances than the ones with official support. Also, additional services like automatic upgrades will depend on this identifier (for instance, through access to the package repositories with the new version packages). System administrators will be able to check the **subscription state**, and link or unlink the subscription through the EdgeScape control panel.

7.2. Edge Workload Orchestration and Management (EdgeNebula)

OpenNebula is an open source management platform to build IaaS private, public and hybrid clouds. It orchestrates compute, storage and network resources to provide a uniform management layer for VMs and infrastructure containers in a multi-tenant environment. EdgeNebula is a specialized version of OpenNebula that incorporates components and enhancements needed to meet the requirements of edge infrastructures. EdgeNebula is therefore responsible for orchestrating the edge infrastructure resources described in Section 6, and managing the life-cycle of the application instances running on the edge infrastructure. EdgeNebula is integrated into the **EdgeProvision**, which registers or unregisters the cloud infrastructure services deployed across the edge locations, as well as, **EdgeScape** which ensures all components and parts are upgraded consistently.

Figure 7.5 depicts the main components of OpenNebula along with the main enhancements needed to extend its functionality to manage highly distributed edge infrastructures. In this section we briefly describe these components and analyze the extensions needed.

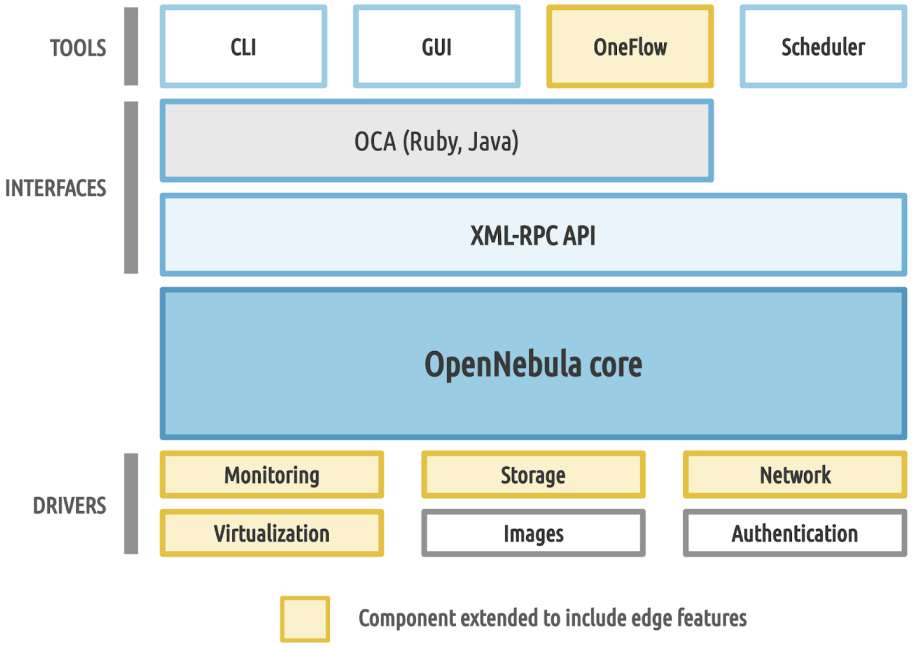

Figure 7.5. OpenNebula architecture and main components.

7.2.1. Host Management

Edge nodes are managed as OpenNebula hosts. These hosts are distributed across different networks and accessed through the public internet and unreliable connections. This model deviates from the traditional private cloud one based on on-premise resources interconnected through private, high-performing networks.

In order to support the previous scenario the following enhancements are required:

- Secure communication in the control plane. Every operation on the edge nodes needs to be initiated through a secure, authenticated channel.
- Monitor information needs to be sent encrypted to the OpenNebula front-end.
- The monitor channel needs to include some reliability mechanisms to control connection errors, at both the transport and application layers.

The use of edge infrastructures will potentially increase the number of managed entities by the OpenNebula front-end. In order to preserve its current performance levels, it is required to decouple the processing of the monitor messages from the main workflow of the OpenNebula daemon.

Note that in any case the edge resources should be able to operate independently in the event of losing a connection. EdgeNebula will provide the mechanisms to monitor the health, status trigger alert alarms, and perform recovery actions in such situations.

7.2.2. Virtualization Management

OpenNebula currently includes virtualization drivers to interact with KVM, LXD containers and vCenter. EdgeNebula will also feature support for AWS's Firecracker hypervisor in order to support scenarios where light micro-VMs are needed. Firecracker is an open source hypervisor based on the KVM virtualization technology. It is specialized in providing secure and light-weight micro-VMs tailored for serverless applications. Firecracker also includes the security features and tools to execute the micro-VMs in a multi-tenant environment. These characteristics make Firecracker an ideal technology to support a broad range of edge applications.

Apart from the support of a new hypervisor, the VM and container management model will be extended to allow better backup and snapshotting functionality that will help to address some of the faulty characteristics of the edge environments.

7.2.3. Network Management

The network drivers of EdgeNebula will include additional functionality to implement the network model described in Section 6.3. In particular, each provider will need one or more specialized components for:

- IP address management (IPAM) drivers to interface the provider network provision API. Typically these are required to register public address ranges to an edge allocation
- SNAT and DNAT support for the network drivers to route inbound and outbound application traffic
- Specific action to install routing rules in the provider to route the application traffic to the edge node

Some network-specific use cases require the deployment of virtual network functions (VNF) on the edge. In these scenarios it is extremely important to optimize the performance of the application instances. Usually, the optimization requires fine-grained, NUMA-aware placement

of the instances and the use of specialized network toolkits like the data plane development kit (DPDK). EdgeNebula will support these high performing edge application instances.

Finally, OpenNebula secures the application instance traffic through simple rules (security groups) that define the allowed inbound and outbound network packets. EdgeNebula will extend the current support for security groups to the edge hypervisors considered, in particular for the NSX vCenter component.

7.2.4. Storage Management

OpenNebula provides a storage abstraction layer that enables it to use multiple storage backends and seamlessly use them in a distributed set of hypervisors. EdgeNebula will extend this abstraction layer to the edge infrastructure by developing specialized image datastores.

The edge image datastores will be able to synchronize the front-end image repository to the edge nodes, so it is completely transparent for the user where the application is being deployed. The synchronization will occur through public networks so secure transfer distribution mechanisms will be used.

The synchronization will be guided by the administrator of the edge infrastructure by selecting the application images that should be available at edge locations.

7.2.5. Multi-Instance Applications

OpenNebula features OneFlow, a multi-VM application (flow) orchestration component. OneFlow coordinates the deployment of applications that include multiple VMs with deployment dependencies and auto-scaling rules.

This feature is quite relevant for deploying edge applications like serverless and container orchestration platforms that involve multiple application instances. EdgeNebula will build upon this functionality extending the management capabilities of the flows:

- Dynamic creation of virtual networks to interconnect flow applications
- Ability to update the auto-scaling rules of a flow
- Re-design of the flow control mechanisms to enable a better scalability

Finally, EdgeNebula will extend the definition of the Flows to be able to import them from the application marketplace.

7.3. Edge Provider Selection (EdgeCatalog)

The EdgeCatalog is a new component that maintains a list of edge resource providers which are certified to work with ONEedge. This component will be created anew in the context of the ONEedge project. This section specifies the data model of the catalog as well as how the user is presented with the information. Also, a specification is given for the process of enrollment of new providers in the EdgeCatalog, as well as the needed guides to be developed to convey this process.

This catalog is centralized and managed by OpenNebula Systems. The catalog includes bare-metal offerings from cloud providers, which can be provisioned into Edge Platforms by the EdgeProvision component. The first iteration will include locations from Amazon EC2, as well as, Packet.

One of the strategic goals of the project is to create a community of providers around this catalog. This community is an ideal vehicle to strengthen the position of ONEedge in the edge platform arena. This positioning will achieve a positive feedback loop where edge providers will

apply to be a part of the catalog since it will improve their exposure to potential customers. Partnerships between OpenNebula Systems and the edge providers can be defined in order to highlight or prioritize their offerings within the catalog, and offer promotions, discounts and other tools to stand out from other providers.

7.3.1. Edge Provider Catalog

The EdgeCatalog data model captures the information needed to decide which provider offers the characteristics better suited for the edge application in terms of capacity, latency, bandwidth, etc.

Each provider has an entry in the catalog with:

- Name
- List of locations, each location with a list of instance types
- Each instance type has:
    - Name
    - Capacity
    - Price per hour
- Price per megabyte (outbound and inbound)
- Link to ONEedge EdgeProvision drivers
- Additional characteristics
    - Public IP
    - SSD disks
    - Bandwidth
    - GPUs
    - Latency wrt ONEedge core services

The catalog is available through the web interface, as well as, through a command line interface. The data model is stored in yaml. The drivers to interact with the different providers are linked from the metadata of the providers, and if they are not present in the particular ONEedge instance connected to the EdgeCatalog (due to them not being available at the time the ONEedge core component as deployed, or a new version is available), they can be added to ONEedge using a one click interface.

Additionally, a set of tools is available to filter the catalog using criteria useful for edge application deployments. The following tools are envisioned:

Latency Calculator Filter

Given a geographic location and a latency threshold, this filter will select those locations that ensure a latency below the given threshold. The filter will work using heuristics that approximate the value according to periodic tests run by OpenNebula Systems among the locations available in the catalog.

Cost Calculator Filter

Given a price target per hour and a capacity in terms of available memory and CPU, this filter selects those locations and instance types that will meet the price objective when deployed.

Specific Characteristics Filter

Given some attributes from a predefined set (those available in the Catalog metadata, for instance, SSD disks in the instances) this filter selects those locations and instance types that meet this criteria.

7.3.2. Certification of New Providers

In order to certify a new provider, an “Edge Provider Certification Guide” will be published covering the process of registering and certifying a new provider in the EdgeCatalog. Also a “Edge Provider Driver Development Guide” will describe the architecture of the EdgeProvision drivers needed to interact with the Edge Provider API programmatically.

The Edge Provider Certification Guide will describe the following processes:

- **Engage**. Initial contact between the edge provider and OpenNebula Systems. Links to the relevant guides sent to the edge provider describing the process and helping with the next steps.
- **Development**. The Edge provider develops an EdgeProvision set of drivers and provides the metadata to add an entry in the EdgeCatalog.
- **Certification**. OpenNebula Systems runs acceptance tests with an account provided by the edge provider. This account needs to be open throughout the lifetime of the EdgeCatalog entry, since the acceptance tests will be part of the certification process of each release. An estimation of the frequency of these tests will be provided in the Certification Guide.
- **Release**. Addition of the provider to the EdgeCatalog. An announcement in the social media instruments will be made. Potentially, a partnership is defined with the edge provider.
- **Support**. Ongoing maintenance and support of the EdgeProvision drivers by the EdgeProvider.

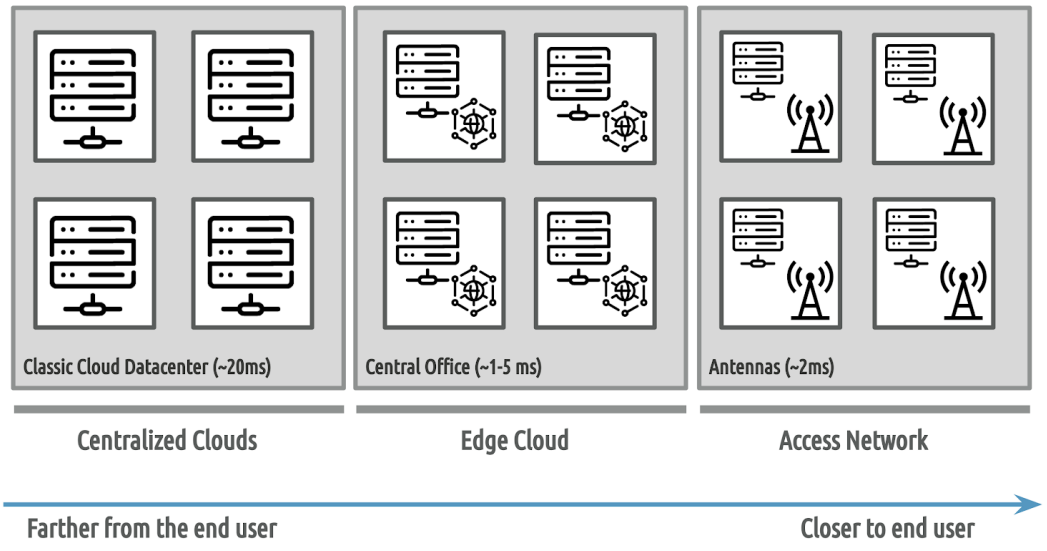

Figure 7.6. Different types of Edge Resource Providers by proximity to their end users.

7.3.3. Types of Edge Resource Providers

The edge computing field is blooming at an accelerated pace. Hence, new edge providers are publishing their offerings continuously. We have identified the following types of Edge Providers that can be represented in the EdgeCatalog, based on the latency they are able to deliver to end users.

Centralized Cloud Providers

This offering includes public cloud providers with a bare-metal offering. As opposed to their classic cloud provider offering (ie, the ability to manage virtual machines) the bare-metal offering opens the possibility to define different edge platform architectures. For ONEedge, this means being able to offer a common architecture for Edge Platforms across different public cloud providers. They typically can offer a latency for edge services above 20 ms.

Examples of centralized public cloud providers with bare-metal offerings include classic public cloud providers like Amazon AWS, Google Cloud Engine, IBM Cloud; as well as small infrastructure providers like Packet, Linode, Scaleaway, and many others.

Edge Cloud Providers

This type encompasses any company with real estate within a population, housing servers that can be consumed through an API. The greater proximity to the end user (compared with a centralized cloud) enables a lower latency for the edge application, in the order of 1-5 ms.

Examples of edge cloud providers are traditional Telecommunication companies through the CORD model (Central Office Re-architected as a Datacenter) like, for instance, Telefonica, AT&T, Deutsche Telekom and many others. Also, this category includes infrastructure providers with presence in different population centers, like, for instance, well-established companies like Equinox as well as members of the Kinetic Edge Alliance (Linode, Packet); and new entries in the cloud provider arena, including Walmart.

Access Network Providers

This category aggregates providers with specialized antennas able to house computing resources. This renders the closest computation to the end user, and can deliver latencies in the order of 2ms.

Some members of the Kinetic Edge Alliance fit into this category (like for instance Vapor IO) as well as specialized companies like Smarter City Technology. Telecommunication companies are also in a good position to act as access network providers.

7.4. Edge Infrastructure Provision and Deployment (EdgeProvision)

The Edge Infrastructure Provision and Deployment (**EdgeProvision**) is the component based on CLI tool OneProvision extended for edge to meet the UX, usability, flexibility and improved reliability required in a highly distributed edge environment. EdgeProvision allows users to manage the full life-cycle of the complete independent edge locations, starting with their provision and maintenance, until the unprovision. This component is integrated into the **EdgeNebula** to easily control the edge locations from GUI, as well as with **EdgeScape** maintenance features to ensure all components and parts are upgraded consistently.

Each edge location (the “**provision**”) is defined as a group of physical hosts allocated from the remote bare-metal cloud provider. They are fully configured with the user-selected hypervisor and enabled in the edge stack for the end users. Except for the physical hosts, each provision comes with dedicated virtual networks and datastores (as described in sections 6.3.2 “Network

Model" and 6.3.3 "Storage Model"). Every single edge location is an independent and complete computing environment.

Location provision goes through the following phases:

- Create **location specific entities** in EdgeNebula.
- **Hosts deployment** when it allocates hosts from remote cloud providers.
- **Hosts configuration** to comply with the reference **Edge Cloud Architecture** (as described in Section 6).
- **Enablement** of edge location for end use.

With the following components involved in the process:

- **Host provision drivers** which allocate / release hosts from remote cloud providers.
- **Configuration specification and application** of host configuration steps.
- **IP address management drivers (IPAM)** to reserve IP addresses for VMs.
- **Network drivers and helpers** to ensure reserved addresses are routed to their VMs.

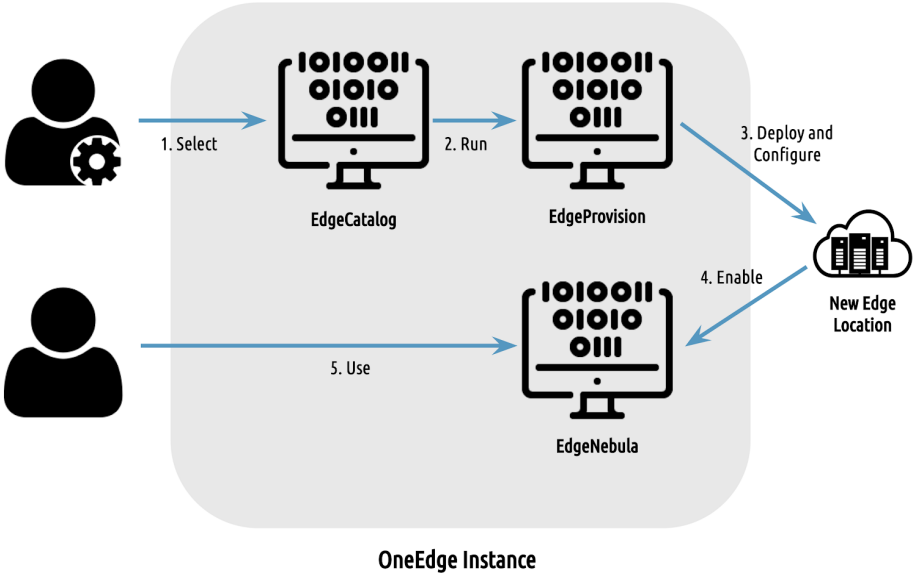

Figure 7.7. Overview of edge location deployment steps.

The deployment of an edge location follows a detailed specification for all components involved. Such specification (a "**provision template**") is internally created based on selections from the EdgeCatalog (e.g. provider, location, or hardware types) and other extra parameters provided by the user (e.g. access credentials, deployment sizing), and passed to the **EdgeProvision** deployment and provision internal logic.

7.4.1. Automatic Provision of Edge Resources

The deployment and provision of edge resources is going to be powered by the CLI tool OneProvision. It is the **provision engine**, which takes care of all necessary steps to bring up a brand-new ready-to-use location, as well as to maintain and to unprovision it at any time cloud administrator decides. The tool encapsulates all the complexity into a single command

execution, with the only notable drawback in the time required to finish the task. The tool should provide visibility into the executed process and its state.

Any non-GUI features related to edge locations management will be implemented in the edge-enhanced version of OneProvision or by the components directly triggered by the tool (as described in overview of section 7.4).

The end user interaction with the edge provision engine is going to be provided via a **dedicated GUI section** within the EdgeNebula interface. The CLI tool OneProvision stays as a hidden engine inaccessible to the regular users. Figure 7.8 demonstrates the wireframe of EdgeProvision integrated into EdgeNebula GUI with a new tab for edge locations and details.

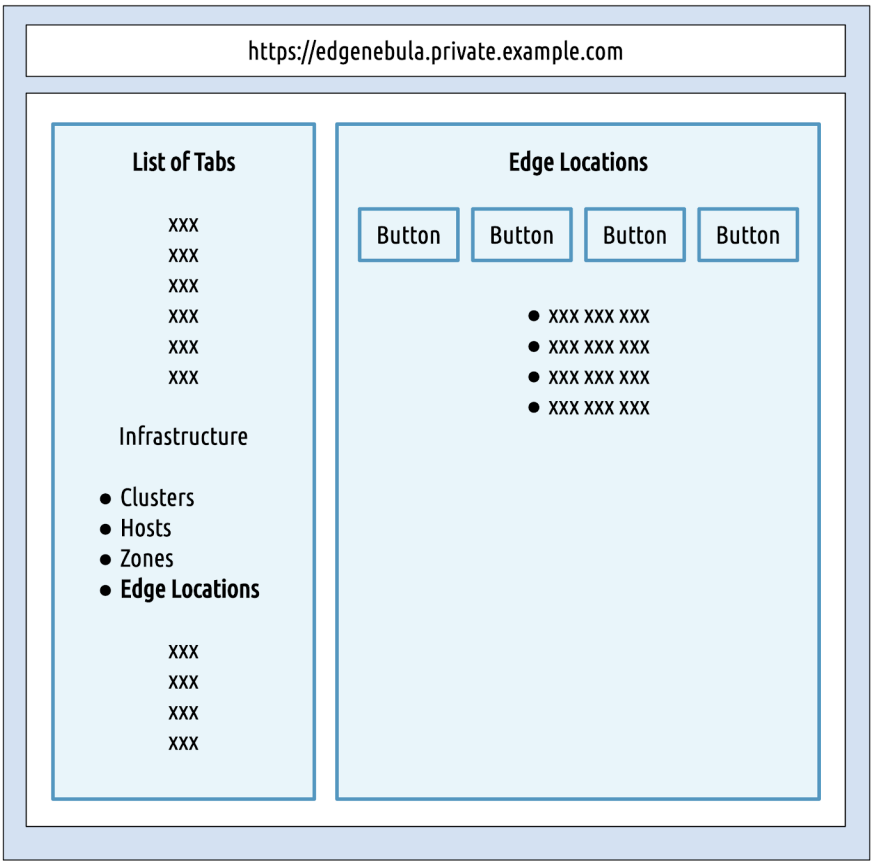

Figure 7.8. Wireframe of EdgeProvision integrated into EdgeNebula GUI.

Edge Location Management Operations

The provision engine and GUI will provide following operations with the **edge locations**:

- **Create** new edge location
- **Delete** edge location
- **List** existing locations and show details
- **Configure** to reconfigure services on the edge hosts
- **Scale-in/out** to change sizing of existing location (ad-hoc add or remove hosts)

It should also allow additional maintenance operations with individual hosts, for example:

- **Power-off** and **power-on**
- Reboot
- Reset

Provision Phases

As described in the overview of section 7.4, the deployment of new edge locations goes through several phases. Each phase of initial and final location lifecycle operation is described below in more detail.

- **Provision** (creation) of new location goes through following phases:
    - Create **location specific entities** in EdgeNebula. Each edge location in the EdgeNebula is implemented as a group of physical hosts with virtual networks and datastores (see section 6 “Distributed Edge Cloud Infrastructure Design”). All these entities form a compact, stand-alone location (cluster) with no additional dependencies. Creation of the entities in the EdgeNebula instance de-facto establishes the new edge location in the inventory.
    - **Hosts deployment**. The provision engine triggers the remote cloud-specific drivers to allocate new edge hosts. Hosts are provisioned in parallel (with a configurable limit). The next phase continues when all requested hosts are ready.
    - **Hosts configuration**. The hosts allocated from the remote cloud provider are expected to be running only a minimal operating system, without any specific services preinstalled. Software components required by the Edge Cloud Architecture are going to be installed, configured and started in this phase. The provision engine integrates with the popular configuration management system (CMS, e.g. **Ansible)** to define and enforce a desired system configuration state. Parallelization of configuration is handled directly by the CMS.
    - **Enablement**. At the end of successful provision, the edge location is enabled for end use.
- **Unprovision** (delete) of existing location through following phases:
    - **Terminate VMs and delete images**. If location is still actively used, the unprovision is not (by default) allowed. The operation can still be enforced by request, to drop the current running state in the location.
    - Switch **hosts into maintenance** mode to prevent starting new VMs on the location.
    - **Undeploy hosts**. Hosts allocated from remote cloud providers are released back.
    - **Delete location from EdgeNebula evidence**.

Error Handling

Location management operations encapsulate the complex logic, which is expected to be **time consuming** (e.g. tens of minutes) and **prone to errors** (e.g. depends on network connection

reliability and remote cloud providers API availability). The provision engine must expect failures and deal with them by:

- **Retrying** operations which can be safely retried.
- **Cleaning up** (rollback) all changes in case of failure.
- Implementing an **asynchronous cleaning** mechanism for fatal failures.

Combination of all approaches must be possible.

7.4.2. Provision Drivers

Integration with remote cloud providers is going to be provided by a set of specialized drivers.

- **Host provision drivers**. These allocate physical resources on the edge location over the remote cloud provider's API and expose basic host state management operations (power-off/on, reboot, reset). At the end of edge location lifecycle, the hosts are released back to the provider.

  A provisioned **host** is represented as a virtualization host in EdgeNebula.

- **IP address management drivers (IPAM)**. These reserve the pool of IP addresses from the provider's pool and assigns them to the VMs. Without such agreement, the (public) networking could not be possible in a fully automated way as the majority of providers are not expected to allow the use of custom random IPs without a previous deal.

  A reserved pool is represented as an **address range** of a **virtual network** in EdgeNebula.

- **Network Drivers and Helpers**. A set of network integration tools and drivers which help with assignment of IPAM agreed addresses to the running VMs. If necessary, the helpers are expected to notify remote cloud providers and request routing of a specific IP from reserved pool to particular physical host and running the VM.

The challenging point is not only to have an edge location which is able to run user workloads, but also be integrated within the remote cloud provider infrastructure on the network level (via the IPAM and network drivers). It is the approach on how to deal with potential restrictions which the remote infrastructure introduces and provide the transparent usability to the end users, regardless of the selected remote cloud provider.

7.4.3. Provision Specification

The deployment of an edge location is controlled by the provision specification document, the **provision template**. It covers all details and parameters of each provision phase, mainly:

- Type of **edge cloud architecture** to configure on hosts
- List of requested **hosts** - what, where and how are going to be provisioned
- List of essential **location specific entities** with parameters for EdgeNebula
  - datastores
  - virtual networks with IP address range
  - virtual network templates

The component will be shipped with a set of partial base provision templates which will define supported edge cloud reference architectures and configurations. A suitable base provision template is going to be provided based on the Infrastructure Administrator's provision selections in the EdgeCatalog.

The final provision specification will be internally generated with information merged from:

- **Base provision template** with comprehensive deployment architecture.
- **User provided inputs** (e.g. credentials, sizing, hypervisor type)

It will then be passed on to the provision engine for a new edge location to be deployed.

7.4.4. Hosts Configuration

Hosts configuration logic is delegated on the configuration management system (CMS) Ansible (currently) integrated with the component. The type of edge cloud architecture to configure on hosts is still specified in the provision template (see 7.4.3), but the particular configuration steps to apply are described in formats and ways specific to **Ansible**, including:

- **Ansible Roles and Modules** to describe steps (tasks) to manage isolated components (e.g. configuration of Linux bridges, installation and management of KVM hypervisor, or SSH configuration and keys exchange)
- **Ansible Playbooks** to describe complete configuration flow as a sequence of applications of several Ansible roles, modules and tasks.

The Ansible playbooks contain both what and how must be configured on hosts to follow the Edge Cloud Architecture. They will be shipped with the EdgeProvision component and hidden to the user behind the selections in the EdgeCatalog UI. Cloud administrators can modify the configuration only via UI through the set of **configuration phase tunables** (e.g. hypervisor to install), which are passed through the final generated provision template into the configuration phase.

While the provision template is a high-level deployment specification of edge location, the configuration playbooks are a low-level description of each step directly executed on the newly allocated hosts.

7.5. Edge Apps Marketplaces (EdgeMarket)

The **EdgeMarket** component aggregates all the sources of virtual machines and container images plus the needed metadata. Hence, it renders a marketplace of appliances that are ready to be deployed using the ONEedge Platform.

The ONEedge interface pulls information from the EdgeMarket component to present an application catalog to the Application Administrator. The data model of the EdgeMarket entries extends the current OpenNebula public marketplace appliance entries, adding complete multi-VM applications as first class citizens. Currently, only single VMs with one disk are supported.

- Name
- Version
- Description
- Logo
- Networks
- Roles:
    - OpenNebula Template
    - OS
    - Cardinality

- ElasticityRules
- Images:
  - Format
  - Driver
  - Size
  - Checksum

All existing and new marketplaces created in the context of ONEedge will implement the above data model. Current marketplaces will be extended to also conform to this data model.

7.5.1. OpenNebula Marketplace

The existing OpenNebula public marketplace[5] contains a selection of appliances that can be deployed into an OpenNebula cloud. It exposes an HTTP REST API, that will be used as the basis for the common marketplace API that needs to be implemented to be a valid source of the EdgeMarket component.

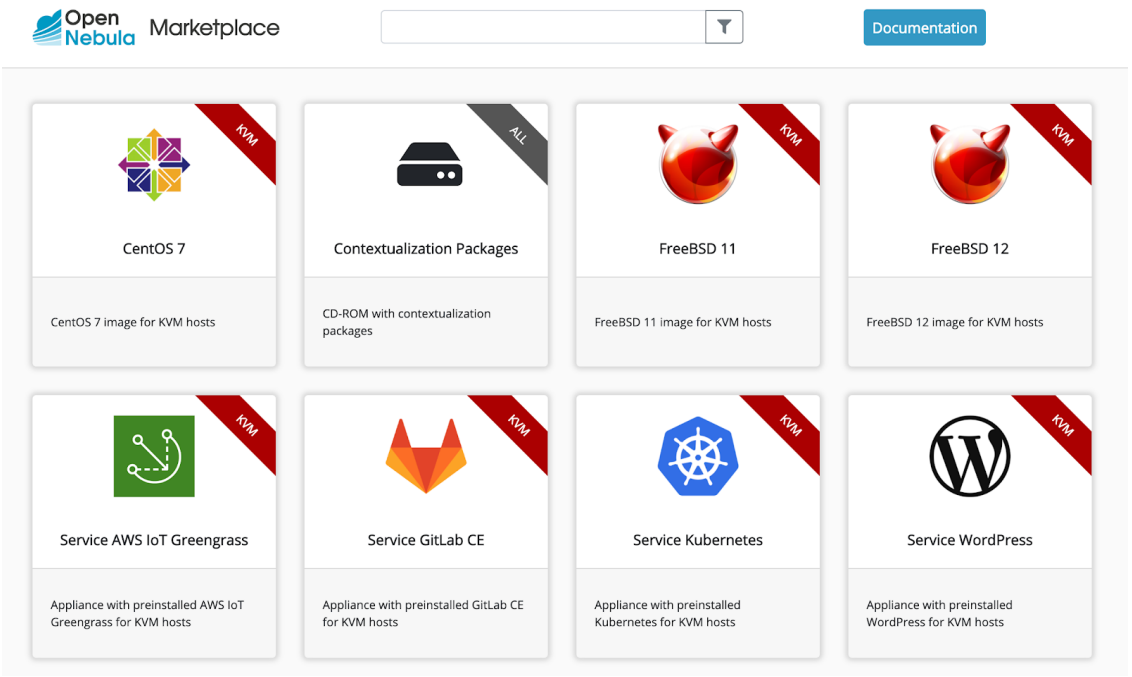

Figure 7.9. OpenNebula application marketplace ready for use.

The appliances currently distributed in the OpenNebula Marketplace contain applications which will benefit from a deployment on the edge. Such appliances includes:

- Amazon GreenGrass for IoT devices
- Kubernetes
- Wordpress
- GitLab

[5] http://marketplace.opennebula.io

The existing marketplace will be extended to overcome existing limitations:

- No service (OneFlow Templates) definition
- No VMs with more than one image

Part of the ongoing maintenance of the marketplace is the support of the appliances, keeping them updated for newer versions of the software. This will be a key part of the work to be carried out on the OpenNebula marketplace.

Moreover, additional popular applications suitable to be run on the edge will be added to the marketplace. Examples of applications to be added to the marketplace are K3s (a lighter version of Kubernetes), Artificial Intelligence and Machine Learning frameworks (TensorFlow, Spark), distributed DBs (Cassandra) and other applications that require low latency or are used as components of more complex applications that are going to be run in the edge.

7.5.2. Container Integration

System containers provide isolated environments without the overhead imposed by virtualization technologies. ONEedge will aggregate at least the following public system container marketplaces to enable the swift and flexible deployment in the edge infrastructure:

- Linux containers[6]
- TurnKey Linux[7]

Application development in current times is heavily relying, more and more, on microservices architecture, which avoids breaking the whole stack when a particular feature is modified or added to the application. This trend in development is tightly coupled to the way these applications are deployed, usually through **application containers**, and most popularly, through Kubernetes, a container management platform. This is a development and distribution model for applications and is specially suited for edge environments, since only a few legacy applications will ever be deployed on the edge, and new applications will most likely be developed using these modern models.

There will be two different approaches to integrate Kubernetes. The first approach follows the general Edge Application deployment model, where a multi-VM application representing a full Kubernetes cluster, optionally with elasticity rules, is deployed as a single entity on a particular Edge Deployment Cloud location. The second approach implies the deployment of a Kubernetes controller as a managed service, offering the following functionality:

- Automatic updates
- Specific version deployment
- Automatic credentials configuration

Future work will include the management of High Availability for this component. This deployment and management will be fully integrated and available through the ONEedge component. Once the managed Kubernetes controller is deployed, adding worker nodes using the multi-VM application concept will be possible on different Distributed Edge Cloud locations (deploying a different role within the multi-VM application representing the Kubernetes cluster on a different location), managing its lifecycle as a single entity.

[6] https://images.linuxcontainers.org
[7] https://www.turnkeylinux.org

In both approaches, ONEedge will enable the simple point-and-click deployment of a complete Kubernetes cluster in edge locations, being able to snapshot and revert the state, suspend and resume the whole Kubernetes cluster as a single entity.

The next step will be to extend the multi-VM support mechanism in EdgeNebula to support hierarchies of multi-VM applications. This will enable the deployment of multiple Kubernetes clusters in different edge cloud locations as a single entity. Guides will be available in the ONEedge documentation (and integrated within the ONEedge interface) to connect to the Kubernetes instances though the command line interface.

7.5.3. Application Container Marketplace

ONEedge will offer access to an Application Container catalog based on the public Helm Chart catalog[8]. This catalog will be accessible through the ONEedge interface, with a slightly different look and feel than the OpenNebula Marketplace to show that containers will be deployed rather than VMs.

However, the same mechanism will apply to deploy a container from the marketplace, with the same workflow as the deployment of single or multi-VM applications. At the time of selecting the location, a list of suitable Kubernetes instances deployed through ONEedge will be presented, and the Helm chart will be automatically deployed in said Kubernetes instances. This will provide basic management of Application Containers. More advanced functionality and management will be available using the Kubernetes command line interface.

[8] https://hub.helm.sh

8. Software Requirements

This section identifies the software requirements and functionality gaps in the main components of the ONEedge architecture, described in Section 7, derived from the user requirements, defined in Section 4.

8.1. Edge Instance Manager (CPNT1)

SR1.1. Simple Product Deployment
Description: Simple deployment of ONEedge stack based on extended OneScape. ● Edge stack installation support in OneScape. ● Support dependencies installation. ● Support operating system (re)configuration. ● Validation steps (pre/post actions). ● Implement a simple installation automation script (with miniONE-like UX).

SR1.2. Automatic Product Upgrade
Description: Upgrade existing deployment based on extended OneScape, fine-grained installation and upgrade of edge stack components. ● Support local services upgrade (edge stack, dependencies). ● Support edge locations (e.g. non-frontend nodes) upgrade. ● Implement rollback to initial state on failure. ● Extra validation steps (pre/post actions). ● Abstraction layer to control OS entities (e.g. package, service, repo). ● Safe services shutdown.

SR1.3. Instance Management
Description: Instance management features to configure, control and monitor state and edge stack and selected system services on the ONEedge host. ● Via set of preselected options provide a way to reconfigure services. ● Access systems logs, integrate with debug bundle generator. ● Support single-click upgrade of ONEedge host operating system. ● Implement a built-in services monitoring and service control. ● Backup and restore support.

SR1.4. Subscription Management
Description: Provide a way to identify, track and manage subscription state of deployments. ● Define and implement mechanism to uniquely identify EdgeNebula instances. ● Show subscription state in EdgeScape control panel. ● Gather instance ID when generating debug bundle. ● Implement subscriptions inventory - evidence of clients' subscriptions.

SR1.5. Web Control Interface (GUI)

Description: Web based interface to control all EdgeScape features.
- New web interface.
- New backend server.
- Support asynchronous jobs processing.
- Implement means to operate control interface from scripts.

8.2. Edge Workload Orchestration and Management (CPNT2)

SR2.1. Integration with Serverless Hypervisor

Description: Implement support for light micro-VMs. Micro-VMs offers a complete isolated environment with a reduced virtual hardware and feature set. This hypervisor is of special relevance for serverless and function-as-a-service scenarios.
- Implement virtualization drivers to support Firecracker.
- Extend the network and storage drivers to accommodate Firecracker instances.
- Add support for VNC connections.
- Add sample VM images and kernels to the Marketplace.

SR2.2. Specialized Cache Datastore

Description: Extend the OpenNebula datastore model so images can be cached to local storage areas in the nodes.
- Image data model needs to include cached locations.
- Image API expose cache operations (e.g. commit changes).
- Develop file-based drivers to transfer images from the datastore to the cache datastore.
- Adapt file-based drivers to be able to use the cache storage area.

SR2.3. Secure and Scalable Distributed Monitoring

Description: Support for distributed edge nodes accessed from public networks.
- Use a secure communication channel for the edge node data (monitoring) and control (application instance) plane.
- Scalable architecture to support large size deployments.
- Segregate monitor traffic in different messages with custom update frequencies.
- Provide a mechanism to easily subscribe third party applications to alarm and warning states.
- Implement a reliable mechanism to send monitor messages.

SR2.4. Virtual Machine Management Operations: Backups

Description: Extend the VM and container management model to allow better backup functionality.

- Define a backup model for application disk images.
- Develop backup specific drivers to accommodate different storage and backup solutions.
- Extended OpenNebula periodic actions to perform backups.

SR2.5. Integration with Remote VMware vCenter Service

Description: Integration with remote cloud hosted VMware service (e.g. VMware on AWS) including NSX and security groups functionality.

- Define the mapping between NSX semantics and OpenNebula semantics.
- Extend the vCenter network drivers to support NSX.

SR2.6. VNF Support

Description: Support for high-performance application instances that implements VNFs.

- NUMA-aware edge node data model.
- NUMA-aware placement algorithms.
- Support for DPDK in Open vSwitch bridges.

SR2.7. Support for Flows in Marketplace

Description: Extend the flow definition model to be able to import complete flow (multi-apps instances).

- Extend the marketplace data model to support multi-template applications and flow definitions.
- Extend the marketplace drivers to import and export flows.
- Extend the associated GUI interfaces.

SR2.8. Complete Service Flows

Description: Extend OneFlow to support additional flow operations as well as to better scale.

- Manage (create and destroy) virtual networks associated with a flow.
- Include update operations to flow roles and auto-scaling rules.
- Replace the current polling mechanism with an event driven approach to update flow states.

SR2.9. Web UI extensions

Description: The functionality developed in SR2.1 - SR2.7 will require adaptations to current interfaces to include new options or improve its layout based on the new data model.

- OneFlow tab to support virtual network definition.
- MarketPlace tab to support multi-VM applications.
- Host tab to include NUMA and Firecracker related options.
- Virtual Networks tabs to include NSX and DPDK.
- VirtualMachine tab to display disk backups and policies.

8.3. Edge Provider Selection (CPNT3)

SR3.1. Edge Provider Catalog Service

Description: Implement a backend to persist the information of the available edge providers.

- Implement the data model persistence.
- Implement an API to manage edge provider entries.
- Implement automatic driver install and configuration in local ONEedge instance from the public Edge Catalog.
- Add Amazon EC2 and Packet entries.

SR3.2 Edge Resource Latency Calculator Filter

Description: Filter providers according to a latency threshold w.r.t a geographic point.

- Add monitoring feature to the provider catalog service, to measure latencies from a bare-metal instance within a defined area centered on the provider location.
- Define heuristics to approximate latencies in a geographic area based on the metrics from the previous point.

SR3.3 Edge Resource Cost Calculator Filter

Description: Filter providers according to maximum cost per hour.

- Add price per hour per instance type information to the catalog data model.
- Develop algorithm to filter out bare-metal instances which cost is higher than the provided threshold.

SR3.4 Driver Maintenance Process

Description: New third party providers need to have a process to develop, test, contribute and certify its drivers with new versions of ONEedge.

- Improve integration guides for host, network and IPAM drivers.
- Develop driver architecture and ready-to-use driver skeletons that eases development.
- Provide a simplified local-testing framework (DDK).
- Define an acceptance and certification process for each driver type.
- Create development and process guides to foster development within the ecosystem.

SR3.5 Edge Catalog Web Interface
Description: Simple, point and click web interface to filter and select the optimal Edge Provider from the catalog. ● Develop and easy to use provider catalog access through a web interface. ● Available standalone and embedded in the ONEedge main interface with no changes. ● Offer provider filter capabilities.

8.4. Edge Infrastructure Provision and Deployment (CPNT4)

SR4.1. Reliable Edge Resource Provision
Description: Improve OneProvision reliability. ● Automatic retry and clean-up operations. ● Background detection and clean-up of orphaned hosts. ● Progress state detection and reporting. ● Testing scenarios maintainability and reliability.

SR4.2. Usability, Functionality and Scalability of Provision
Description: Enhance OneProvision functionality and scalability. ● Edge location update (scale-out/in). ● Support new types of entities created during provision (e.g. virt. net. templates). ● Review location and (reduce) content of provision metadata persisted in EdgeNebula. ● Extend EdgeNebula accounting for hosts. ● Packaging of base dependencies.

SR4.3. Provision Template for Reference Architectures
Description: Enhancements to templates describing the reference deployment. ● Ansible roles and playbooks for reference architectures. ● Provision templates for (new) reference architectures / cloud providers. ● Packaging of missing components.

SR4.4. Inter-edge Networking Deployment Scenario
Description: Enhancements to provision and components to support transparent secure connection among geographically distributed edge locations. ● Provision updates to support transparent connection among edge locations. ● VNF appliance enhancements for secure routing among edge locations (VPN).

SR4.5. Drivers for Host Provision

Description: Improvements to host provision drivers and their interface.
- Improve logging and progress reporting reporting from drivers.
- Extend types of operations with hosts (e.g. disk attach/detach).

SR4.6. Drivers for IP Address Management

Description: Improvements to IPAM drivers and their interface.
- Extend information provided to the IPAM drivers actions.
- Allow data inheritance from virt. network to AR.

SR4.7. Drivers for Network Drivers and Helpers

Description: Improvements to network drivers and their interface.
- Propagate network helpers failures to the VM state.
- Deal with MAC/private IP collisions among different virt. networks.
- Isolate host network services reachable from within private networks.
- Lower requirements on pre-configurations done within location provision.

SR4.8. GUI for Edge Resource Provision

Description: Integration within EdgeNebula GUI.
- New EdgeNebula provision GUI extension.
- Update EdgeNebula host interface for state control operations (power-off/on).
- Asynchronous background jobs runner.

8.5. Edge Apps Marketplace (CPNT5)

SR5.1. Edge Applications and Services in Marketplace

Description: Extend current OpenNebula VM and container marketplaces to deal with OneFlow services.
- Extend VM Templates in the marketplace to support multiple images.
- Extend the marketplace to support OneFlow Templates.
- Extend the marketplace drivers to support VMs with multiple images and OneFlow Templates.

SR5.2. Built-in Management of Application Containers Engine

Description: Built-in management of Kubernetes clusters.

- Kubernetes managed controller functionality with automatic upgrades and credential provisioning.
- Implement easy worker node addition/subtraction to the Kubernetes cluster, based optionally on elasticity rules.

SR5.3. Integration with Application Containers Marketplace

Description: End users should be able to easily submit Kubernetes apps from popular marketplaces.

- Develop application container marketplace proxying to helm charts official marketplace.
- Integrate Kubernetes controller management in the ONEedge GUI (new EdgeNebula/Sunstone tab).
- Application Container management in EdgeNebula.

SR5.4. New Edge Applications Marketplace Entries

Description: New appliances suitable for deployment on Distributed Edge Clouds should be available in the marketplace.

- Lite Kubernetes (K3s) appliance.
- Cassandra appliance.
- Tensorflow appliance.
- Apache Spark appliance.

SR5.5. Edge Market GUI Developments

Description: End users should be able to deploy both Edge Applications based on VMs and system containers as well as application containers in the Distributed Edge Cloud using a simple point and click web interface.

- Aggregate OpenNebula marketplace for VMs in EdgeMarket.
- Aggregate System Container Linux Containers marketplace.
- Aggregate System Container TurnKey Linux.
- Add functionality to ONEedge interface for Application Containers enabling deployment in selected Edge Provider locations.

9. User to Software Requirements Matching

This section provides the results of the requirement engineering process, which derives system requirements (functional and non-functional) from functional gaps to implement a system that fulfills use case requirements.

The following tables (Tables 9.1 to 9.5) summarize the resulting system requirements and originating user requirements.

Id	Description	Source
SR1.1	Simple Product Deployment.	UR0.11
SR1.2	Automatic Product Upgrade.	UR0.10 UR0.13
SR1.3	Instance Management.	UR0.11
SR1.4	Subscription Management.	UR0.12
SR1.5	Web Control Interface (GUI).	UR0.15

Table 9.1. System requirements for Edge Instance Manager (CPNT1).

Id	Description	Source
SR2.1	Integration with Serverless Hypervisor.	UR0.2 UR2.1 UR4.1
SR2.2	Specialized Cache Datastore.	UR0.3 UR1.2 UR2.2
SR2.3	Secure and Scalable Distributed Monitoring.	UR0.4 UR0.14 UR2.2 UR5.4
SR2.4	Virtual Machine Management Operations: Backups.	UR2.2 UR4.4
SR2.5	Integration with Remote VMware vCenter Service.	UR1.3 UR0.2 UR0.4
SR2.6	VNF Support.	UR1.1 UR5.1 UR5.2 UR5.3

SR2.7	Support for Flows in Marketplace.	UR0.8 UR1.2
SR2.8	Complete Service Flows.	UR0.5 UR2.3 UR5.5
SR2.9	Web GUI Extensions.	UR0.15

Table 9.2. System requirements for Edge Workload Orchestration and Management (CPNT2).

Id	Description	Source
SR3.1	Edge Provider Catalog Service.	UR0.1 UR0.2 UR0.3 UR1.3 UR4.4
SR3.2	Edge Resource Latency Calculator Filter.	UR0.1 UR0.6 UR3.1
SR3.3	Edge Cost Cost Calculator Filter.	UR0.1 UR0.7
SR3.4	Driver Maintenance Process.	UR0.3
SR3.5	Edge Catalog Web Interface.	UR0.15

Table 9.3. System requirements for Edge Provider Selection (CPNT3).

Id	Description	Source
SR4.1	Reliable Edge Resource Provision.	UR0.4
SR4.2	Usability, Functionality and Scalability of Provision.	UR0.4 UR0.5
SR4.3	Provision Template for Reference Architectures.	UR0.4
SR4.4	Inter-edge Networking Deployment Scenario.	UR2.2 UR3.2 UR4.3
SR4.5	Drivers for Host Provision.	UR0.4 UR0.5
SR4.6	Drivers for IP Address Management.	UR0.4 UR0.5

SR4.7	Drivers for Network Drivers and Helpers.	UR0.4 UR0.5
SR4.8	GUI for Edge Resource Provision.	UR0.15

Table 9.4. System requirements for Edge Infrastructure Allocation and Deployment (CPNT4).

Id	Description	Source
SR5.1	Edge Applications and Services in Marketplace.	UR0.8 UR0.11 UR3.3 UR3.4 UR5.5
SR5.2	Built-in Management of Application Containers Engine.	UR0.9 UR2.3 UR3.3 UR3.4 UR4.2
SR5.3	Integration with Application Containers Marketplace.	UR0.9 UR2.3 UR3.3 UR3.4 UR4.2
SR5.4	New Edge Applications Marketplace Entries.	UR0.8 UR2.3 UR3.3 UR3.4 UR4.2
SR5.5	Edge Market GUI Developments.	UR0.15

Table 9.5. System requirements for Edge Applications Marketplace (CPNT5).

PART III. Software Certification and Verification Plan

10. Software Certification

Software Certification is a process consisting of the recommended testing approaches, comprised of the unit, integration, functionality and acceptance testing to validate the individual components and whole system meet the specification and requirements on functionality, quality and reliability. It is based on the certification process of the existing underlying base components, and is extended for requirements of the new components and environments.

The process is going to be vastly **automated** to provide hassle-free and quick feedback on current certification status of the project or selected component. Nevertheless the significant role in the process also lies on the **manual** testing, which will follow the pre-defined test (usage) scenarios to catch the defects not discovered by automation tests.

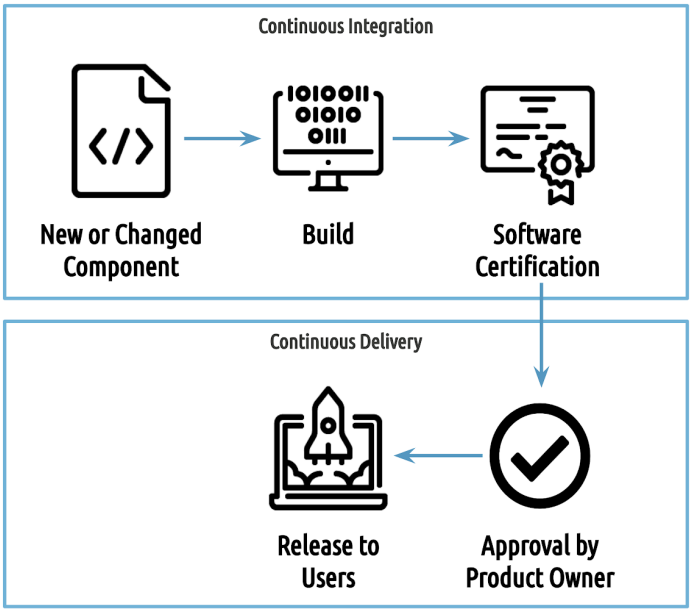

Figure 10.1. Software Certification integrated within complex CI/CD process.

It is a part of the more comprehensive **Continuous Integration / Continuous Delivery** (CI/CD) based process (as visualized in Figure 10.1), which describes all necessary steps to deliver the new product features to the user and even handles the delivery itself.

10.1. Infrastructure

The certification infrastructure will be comprised of two main parts:

- CI/CD automation service
- Certification servers - the computing resources to run the process

CI/CD automation service is going to execute the whole process from build, to certification and delivery to the users. The CI/CD solutions already exist and fit the project needs enough. They

are available as an installable software component (e.g. Jenkins, TeamCity) or in the form of Software as a Service, eventually provided within the code repository hosting (e.g. Travis CI, GitHub Actions, GitLab CI). A combination of different solutions to spread the load and responsibilities may also be suitable.

Certification servers are the biggest part of the infrastructure with growing demand based on the number of tested components, scenarios or certification frequency. The following types of resources will be used for certification:

- **On-premise certification servers**
- External **public cloud certification servers** to
    - Cover temporary on-premise resources shortage
    - Get access to specific platforms (e.g. ARM)
- **Resources from integration parties** (see Section 7.3.2) used by certification servers

Certification infrastructure will depend mainly on the native environments provided by each resource type (e.g. native x86-64 architecture with on-premise resources and native ARM64 from external public cloud provider). Platforms with no flexible way to access or with limited availability can be **fully emulated** on different architectures (e.g. ARM emulated system on x86-64 machine via QEMU) at the cost of degraded certification performance.

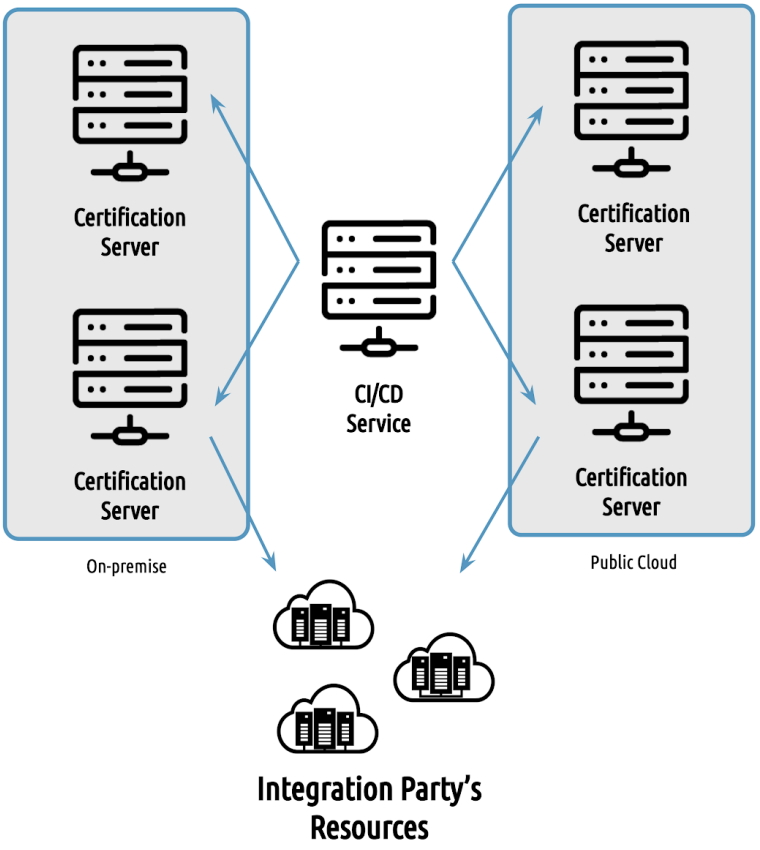

Figure 10.2. Types of resources involved in the certification process.

The CI/CD process workflow (**pipeline**) will follow a programmatic approach, when all steps will be described by a single control descriptor with features of simple programming language. Control descriptor will be version controlled.

10.2. Certification Process

Several components are involved in the certification process, and their combinations form a multi-dimensional **sparse** testing matrix. Not all combinations are expected to be allowed, as some might result in redundant certification or they might not be valid. Example of tested components and their options is in Table 10.1.

EdgeStack Operating System	**EdgeStack** Hardware Platform	**EdgeNode** Operating System	**EdgeNode** Hardware Platform	**Component**	**Version**
CentOS 8	x86-64	CentOS 8	x86-64	edgescape	1.0.0
Ubuntu 18.04		Ubuntu 18.04	ARM32	edgescape-ui	
Debian 10			ARM64	packet-ipam	
				packet-pm	

Table 10.1. Example of tested components and their variants.

The certification process runs tests for each enabled combination and provides a single aggregated **certification report** at the end. Tests might be implemented in frameworks available for each programming language used by particular components. In such a case, test result state must be integrated with the unified **acceptance** (readiness) testing framework to shade the lower-level differences among components.

Preparation of the certification environments, certification execution, reporting and (optionally) delivery of the product to the users are the responsibilities of the introduced CI/CD service.

Certification process can be executed:

- **On event** (e.g. on a code change)
- **Periodically** (e.g. every night)
- **Ad-hoc** (e.g. manually triggered)

10.3. Drivers Certification

While most components will have development and testing fully covered by the company and hidden from the end users, the edge location specific drivers are expected to be developed and maintained by the edge cloud providers. Driver certification will follow the steps described in the “Edge Provider Certification Guide” (see 7.3.2) and although the main certification will be executed internally, drivers developers should provide their own inputs and have visibility into the process.

Each new edge provider integration is going to be represented as a testing scenario in the software certification process (see testing matrix in Table 10.1). Integration can be tested within a single scenario (e.g. scenario “providerName”), or can be split into several ones based on the tested component (e.g. providerName-ipam, providerName-pm, providerName-provision).

Certification process will be flexible enough to handle testing (with optional release) just for the particular component (driver) and not the complete stack.

10.4. Software Delivery and Validation

Certified software is going to be delivered to the users via proposed delivery channels (see 7.1.1) with the semi-automated process. Based on the certification report and other circumstances (e.g. hour, day of week), a responsible **product owner** approves the release of new component(s) to the users.

On approval, CI/CD service executes the automated steps to (for example):

- **Release new component(s)** into delivery channels
- **Update documentation**
- **Validate release**
- **Bump components for next release**
- Or **notify users**

Release validation is an important step to verify all parts were released successfully and are accessible by the end users. For simple delivery steps, the validation will be implemented as an **extra ad-hoc validation queries** on the state (e.g. documentation released over SCP is validated to be available over HTTP). For complex deliverables, the validation involves running a **limited software certification process** (e.g. delivery of new components into software repositories triggers a certification part which installs from the updated repositories and validates versions of installed components).

10.5. Sandboxed Environments

Certification environments provide a defined and consistent initial state for the software certification process, both for automated software certification or manual. Such environments can be created and preserved in their initial state, by skipping the software certification process, and used for:

- **Manual testing**
- **Demonstration** purposes
- **Experiments, development, debugging**

11. Software Verification

11.1. Verification Methodology

The goal of the verification process is to assess that the functional components of the software platform conforms to the requirements identified in Section 8. This, in turns, will validate that the edge computing platform is feature-complete and able to implement the uses cases relevant for the project.

In order to support the agile development adopted in this project, the verification process is integrated with the software certification procedure described in Section 10. The methodology is structure as follows:

- **Verification scenario**. Describes a simple user story that captures one or more functional requirements of a software requirement of a component, see Section 8.
- **Integration scenario**. Describes a complete user story that involves several components. Its goal is to check the integration and interaction of several edge platform components.
- **Verification test**. An automated testing program that exercises the functional aspects of the scenario. Each verification test is then integrated into the certification platform to certificate and test software releases.
- **Integration test**. Similar to the verification test but for an integration scenario.

11.2. Verification Scenarios

This section presents a list of verification scenarios for verifying the software requirements defined in Section 8. Each platform component is presented in a separated table that includes a brief description of each scenario.

SW Req.	Verification Scenario
SR1.1	VS1.1.1. As an Infrastructure Administrator I can install complete edge stack on a dedicated host including dependencies and with necessary operating system reconfigurations managed. VS1.1.2. As an Infrastructure Administrator I can run a pre-installation validation and get a post-installation deployment validation report. VS1.1.3. As an Infrastructure Administrator I can do an installation by running a simplified installator and end-up with working edge stack deployment.
SR1.2	VS1.2.1. As an Infrastructure Administrator I can easily upgrade the deployment. It consistently upgrades edge stack components and edge locations. VS1.2.2. As an Infrastructure Administrator I can execute the upgrade process and in case of failure (e.g. due to network issues), the deployment can be reverted back to the pre-upgrade state.

	VS1.2.3. As an Infrastructure Administrator I can execute the upgrade of the deployment with running actions (e.g. currently deploying VM). The upgrade waits for all actions to finish before continuing.
SR1.3	VS1.3.1. As an Infrastructure Administrator I can reconfigure selected options for selected components. VS1.3.2. As an Infrastructure Administrator I can access system logs and generate a debug bundle. VS1.3.3. As an Infrastructure Administrator I can upgrade the base operating system. VS1.3.4. As an Infrastructure Administrator I can check the health state of the deployed components and control state (stop, start, restart) selected services. VS1.3.5. As an Infrastructure Administrator I can backup state of the edge stack services and restore to the previous compatible backup.
SR1.4	VS1.4.1. As an Infrastructure Administrator I can see my unique deployment identification and its state of support. VS1.4.2. As a Customer Support Agent I can view and control the support state of an uniquely identified customer deployment. VS1.4.3. As a Customer Support Agent I can see an unique deployment identification and support state within debug bundle generated by an Infrastructure Administrator.
SR1.5	VS1.5.1. As an Infrastructure Administrator I have a graphical web control interface which I can use for proposed management operations related to the edge stack components and ONEedge instance. VS1.5.2. As an Infrastructure Administrator I can operate the control interface functions non-interactively from the scripts.

Table 11.1. Verification scenarios for Edge Instance Manager (CPNT1).

SW Req.	Verification Scenario
SR2.1	VS2.1.1. As an Infrastructure Administrator I can register a Firecracker-enabled host and monitor its status. VS2.1.2. As an Application Administrator I can define a Firecracker VM template and perform basic operations: deploy, suspend, resume, terminate and connect through VNC. VS2.1.3. As an Application Administrator I can deploy a Firecracker VM and connect it to public and private networks.

	VS2.1.4. As an Application Administrator I can browse the marketplace and download Firecracker-enabled images and kernels with context packages.
SR2.2	VS2.2.1. As an Infrastructure Administrator I can register a system datastore with a local cache stored area and define which images should be cached. VS2.2.2. As an Infrastructure Administrator I can deploy a VM or container using a cache-able image with negligible deployment times
SR2.3	VS2.3.1. As an Infrastructure Administrator I can monitor hosts through a public link and receive the monitor information encrypted and in a reliable manner. VS2.3.2. As an Infrastructure Administrator I can define different update frequencies for each monitor traffic type. VS2.3.3. As an Infrastructure Administrator I can deploy a large number of hosts (2.5K) and VMs (10K) and monitor them without impact on response times.
SR2.4	VS2.4.1. As an Application Administrator I can define a backup policy for a VM disk VS2.4.2. As an Infrastructure Administration I can define backup datastores to store VM images backups. VS2.4.3. As an Application Administrator I can list the backups available for a VM disk and have a procedure to recover a specific backup.
SR2.5	VS2.5.1. As an Infrastructure/Application Administrator I can define a security group and associate VMs to it. VMs associated with this security group implements its rules using NSX functionality when deployed using VMware vcenter. VS2.5.2. As an Application Administrator I can select a security group to deploy a VM and verify inbound and outbound traffic conform to the rules defined in the security group.
SR2.6	VS2.6.1. As an Infrastructure Administrator I can register a host with a NUMA topology and query it through the host object in EdgeNebula VS2.6.2. As an Infrastructure Administrator I can define a VM with a virtual NUMA topology and a NUMA allocation policy VS2.6.3. As an Infrastructure Administrator I can deploy a VM with a virtual NUMA topology and the VM is placed accordingly its allocation policy
SR2.7	VS2.7.1. As an Infrastructure Administrator I can define a multi-VM application in the marketplace, comprising multiple VM images and roles. VS2.7.2. As an Application Administrator I can import a multi-VM application from the marketplace and instantiate it on the edge infrastructure. VS2.7.3. As an Infrastructure Administrator I can define a multi-VM application that dynamically creates the networks it needs. When the application terminates I can verify that the networks have been removed.

Table 11.2. Verification scenarios for Edge Workload Orchestration and Management (CPNT2).

SW Req.	Verification Scenario
SR3.1	VS3.1.1 As an Infrastructure Administrator I can browse a catalog of different edge providers (Amazon EC2 and Packet at least), accessing information about the location, prices per hour and hardware characteristics. VS 3.1.2 As an Infrastructure Administrator I can programmatically select and install drivers for a edge provider in ONEedge, if they are not present in the system already.
SR3.2	VS 3.2.1 As an Infrastructure Administrator I can filter the edge provider catalog according to a desired latency threshold on a geographic area
SR3.3	VS 3.3.1 As an Infrastructure Administrator I can filter the edge provider catalog according to a cost per hour threshold
SR3.4	VS 3.4.1 As an Edge Provider I can learn how to create new ONEedge drivers for my offering, using integration guides and driver skeleton examples VS 3.4.2 As an Edge Provider I can learn the process to add my drivers into the Edge Provider Catalog VS 3.4.3 As an Edge Provider I can use a Driver Development Kit to perform local testing and thus ensure the correct driver certification
SR3.5	VS 3.5.1 As an Infrastructure Administrator I can select Edge Providers from Catalog using an easy to use web interface, using available filters, through the main ONEedge main interface or a standalone web

Table 11.3. Verification scenarios for Edge Provider Selection (CPNT3).

SW Req.	Verification Scenario
SR4.1	VS4.1.1. As an Infrastructure Administrator I can run an edge location creation. If the process experiences a recoverable problem, it recovers and successfully finishes. Non-recoverable states are automatically fully reverted. VS4.1.2. As an Infrastructure Administrator I ran edge location deployments which experienced fatal non-recoverable error due to which deployment failed. No resources are left long-term running on the remote edge provider out of the edge stack evidence. VS4.1.3. As an Infrastructure Administrator I can see the provision progress state of the new edge location.
SR4.2	VS4.2.1. As an Infrastructure Administrator I can update provisioned edge locations, add a new host(s) and remove existing host(s). VS4.2.2. As an Infrastructure Administrator I can see an approximate money spent for the edge location within the edge stack accounting mechanism. VS4.2.3. As an Infrastructure Administrator, I get reusable network templates for ad-hoc private networks on environments which support that.

SR4.3	V4.3.1. As an Infrastructure Administrator I can deploy to all providers from the EdgeCatalog the proposed reference architectures just by clicking through the wizards.
SR4.4	V4.4.1. As an Infrastructure Administrator I can provision edge locations with transparent cross-edge private networking. VM_1 from one edge location/provider reaches VM_2 from different edge provider/location over dedicated private addresses.
SR4.5	V4.5.1. As an Infrastructure Administrator, I can do advanced host management operations (e.g. attach/detach remote persistent disk/shares) on providers which support them. V4.5.2. As an Infrastructure Administrator, I can watch progress of compound host management operations (e.g. deploy).
SR4.6	V4.6.1. As an Infrastructure Administrator, I can easily allocate extra IP address ranges into existing virtual networks with ARs managed by IPAM without knowledge of low-level infrastructure / provider details and parameters (credentials).
SR4.7	V4.7.1. As an Application Administrator, I can attach or detach public IP (alias) to the VM. A failure during the operation is reflected in the VM state. V4.7.2. As an Application Administrator, I can run VM which can't access internal services provided by the hypervisor host. V4.7.3. As an Infrastructure Administrator, I can have isolated virtual networks with overlapping private IP ranges used by VMs on the same host. Such VMs still support the public IP networking.
SR4.8	V4.8.1. As an Infrastructure Administrator, I can create, show and control the edge locations from a new dedicated tab within the EdgeNebula graphical interface. V4.8.2. As an Infrastructure Administrator, I can control host state (e.g. power-off/on, reset) from the EdgeNebula graphical interface.

Table 11.4. Verification scenarios for Edge Infrastructure Allocation and Deployment (CPNT4).

SW Req.	Verification Scenario
SR5.1	VS 5.1.1 As an Application Administrator I can make use of predefined Virtual Machines available in the Edge Applications Marketplace featuring more than one disk. VS 5.1.2 As an Application Administrator I can make use of predefined multi-VM applications available in the Edge Applications Marketplace.
SR5.2	VS 5.2.1 As an Application Administrator I can deploy a Kubernetes controller as a managed service within ONEedge, or as a regular application. VS 5.2.2 As an Application Administrator I can add/remove easily new worker nodes to my Kubernetes cluster, either manually or through elasticity rules.

SR5.3	VS 5.3.1 As an Application Administrator I can browse the helm chart official marketplace from ONEedge, and deploy applications to my Kubernetes instance.
SR5.4	VS 5.4.1 As an Application Administrator I have access to predefined single or multi-VM applications, including K3s, Cassandra, TensorFlow and Apache Spark.
SR5.5	VS 5.5.1 As an Application Administrator I can make use of an easy to use web interface to deploy applications from at least OpenNebula Marketplace, TurnKey Linux and Linux Containers catalogs. VS 5.5.2 As an Application Administrator I can select in which Edge Provider locations I'm going to deploy my application.

Table 11.5. Verification scenarios for Edge Applications Marketplace (CPNT5).

11.3. Integration Scenarios

11.3.1. Dynamic Provision of Edge Infrastructure

As an Infrastructure Administrator I can query a set of edge locations and select a new location based on its costs and price characteristics. For the selected location two bare-metal nodes are provisioned and the associated services configured. The new servers are added to OpenNebula along with the associated storage (image/system datastores), network and cluster resources.

Then I can instantiate simple VMs in any of the provisioned locations and verify its operational level and network connectivity. Finally I can monitor the status of the overall distributed edge infrastructure.

11.3.2. Dynamic Provision of Edge Infrastructure

As an Application Administrator I can query a catalog of edge applications and select one for importing into an existing edge infrastructure. Then I can deploy the newly imported application on the infrastructure and monitor its deployment progress. Finally I can login in the application and interact with it normally.

12. Conclusions and Next Steps

This document describes the use cases that will guide the development of the project, identifies the main user requirements derived from these use cases, and defines the architecture of the ONEedge management platform. From the user requirements, we have extracted the list of software requirements and functional gaps to be implemented by the components of the ONEedge management platform, and the methods and scenarios to verify their fulfillment.

The new software components and extensions to meet the software requirements will be specified and developed within work package WP3, and the new functionality will be tested, verified and demoed within WP4. Some of the software requirements involve the development of appliances and the automation of infrastructure deployment and configuration that will be performed as well in WP4.

This is the first version of the Solution Framework report that will be updated at the end of each innovation cycle with an analysis of fulfillment of verification tests and scenarios in the cycle and improvements in the architecture and its components.