ONEedge.io

A Software-defined Edge Computing Solution

D4.1. Infrastructure Report - a

Infrastructure Report v.1.0

31 July 2020

Abstract

This report, delivered at the end of the First Innovation Cycle (M4-M9), summarizes the certification infrastructure and processes that we follow in order to verify that software meets the specifications and requirements on functionality, quality and reliability. It also includes a detailed list of the extensions implemented to verify the functionality of the software developed in ONEedge. Tests and extensions of the verification framework are detailed for each component and grouped for each software requirement implemented during the cycle.

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:	WP4. Demo and Operational Infrastructure
Nature:	R: Report
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	31/7/2020
Actual Date of Delivery:	31/7/2020
Lead Authors:	Vlastimil Holer, Rubén S. Montero and Constantino Vázquez
Authors:	Sergio Betanzos, Ricardo Díaz, Christian González, Alejandro Huertas, Jorge M. Lobo, Ángel L. Moya, Jan Orel, Petr Ospaly and Cristina Palacios
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
1.0	31/7/2020	Submitted	First final version of the report

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

Executive Summary

This document provides a summary of the certification infrastructure and processes that we follow in order to verify that software follows the specification and meets the requirements on functionality, quality and reliability. This is an integral part of the software development, and allows it to be executed at any time and state of the product.

Output from the certification is the certification report with both an overall summary and a detailed list of the problematic parts. Based on the certification report, the product source code is iteratively fixed and improved. On successful certification the product can be released through the download servers to the users.

The whole certification process is powered by a certification infrastructure formed by different on-premises and distributed edge servers for automation, build and certification servers, core certification, and download.

During the First Innovation Cycle (M4-M9), the project mostly focused on those software requirements needed to achieve our first milestone in M9, which is the base functionality needed for a single-host edge deployment.

The work carried out during this First Innovation Cycle has involved the software requirements of components CPNT1, CPNT2, CPNT3, CPNT4 and CPNT5, with a special focus on laying the technological foundation of ONEedge.

For each software requirement, this report includes a summary of the extensions performed in the testing and certification infrastructure. Also, for each requirement, we list the verification scenarios that have been addressed and a description of the functionality tested to fulfill the proposed scenarios.

Table of Contents

1. Certification Infrastructure **5**

1.1 Automation Server 5

1.2 Build and Certification Servers 6

1.3 Certification Process 10

1.4 Download Servers 12

2. Software Requirements Verification **13**

2.1 Edge Instance Manager (CPNT1) 13

2.2 Edge Workload Orchestration and Management (CPNT2) 13

2.3 Edge Provider Selection (CPNT3) 16

2.4 Edge Infrastructure Provision and Deployment (CPNT4) 16

2.5 Edge Apps Marketplace (CPNT5) 17

1. Certification Infrastructure

The certification process verifies that software follows the specification and meets the requirements on functionality, quality and reliability. It is an integral part of the development, and allows it to be executed at any time and state of the product. Output from the certification is the certification report with both overall summary and detailed list of the problematic parts. Based on the certification report, the product source code is iteratively fixed and improved. On successful certification the product can be released through the download servers to the users.

The whole certification process (also with steps which precede and follow) is powered by the certification infrastructure (see Figure 1.1). It consists of the following principal parts:

- **Automation Server**
- **Build and Certification Servers**
- **Core Certification Process**
- **Download Servers**

1.1 Automation Server

Follows the Continuous Integration / Continuous Delivery (CI/CD) approach and is backed by popular open source automation service Jenkins (https://www.jenkins.io) deployed on a single Linux server. For a selected state of developed software component it triggers the build and certification process and orchestrates all the parts:

- fetch product code from source code repository
- allocation of computing resources
- product build
- product certification
- reporting
- release artifacts to the downloads servers for users

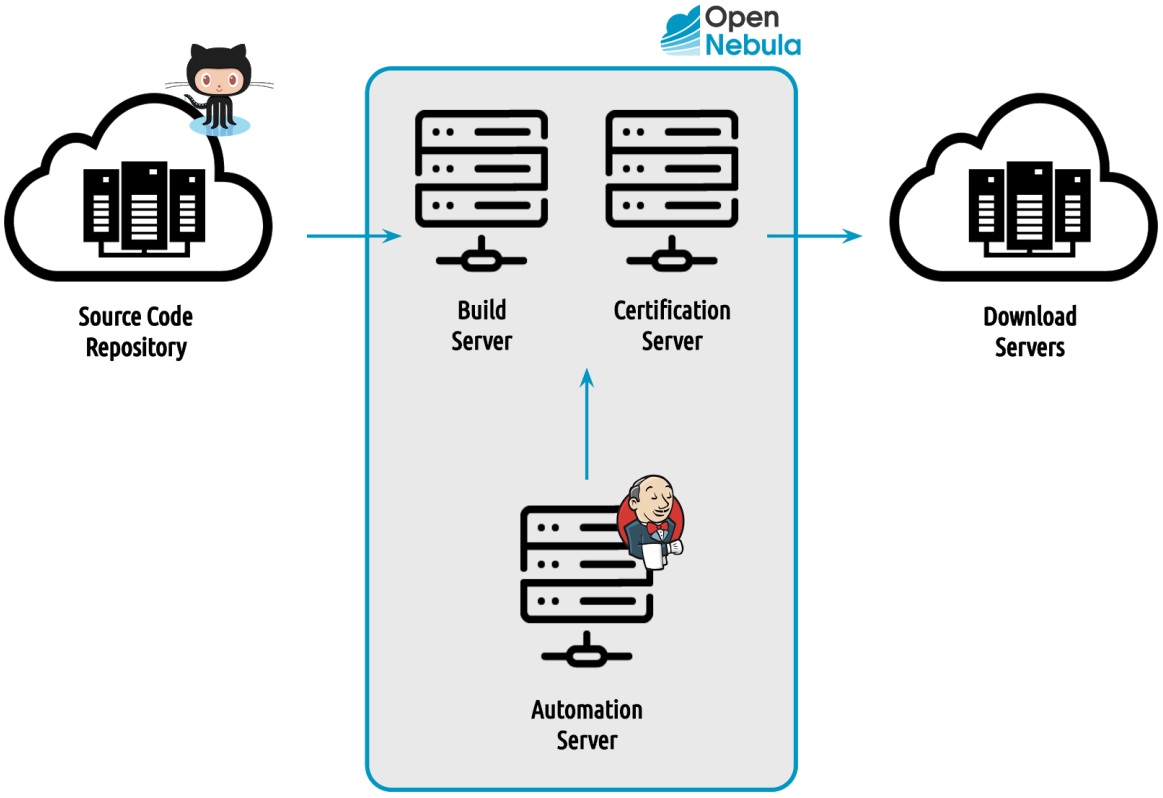


Figure 1.1. Overview of certification infrastructure

The individual steps of the process are implemented as a combination of shell scripts, set of Jenkins Freestyle Projects and Jenkins Pipelines. They are implemented as a source code covered by a history version tracking. The complete certification is automatically scheduled to run nightly for the latest state.

1.2 Build and Certification Servers

Virtualization infrastructure servers provide computing resources for the product build and certification process. They are mainly the on-premise x86-64 bare-metal virtualization servers running KVM hypervisor and controlled by OpenNebula Infrastructure Manager, but can dynamically grow with on-demand edge resources on supported providers which transparently integrate into the Certification Infrastructure.

Simplified infrastructure resource allocation process is displayed in Figure 1.2. After the request from Automation Server (step 1), the Infrastructure Manager allocates the resources (step 2) on virtualization servers and a new group of dedicated Build and Certification Servers (step 3) are started. These are passed back (step 4) to Automation Server for further use.

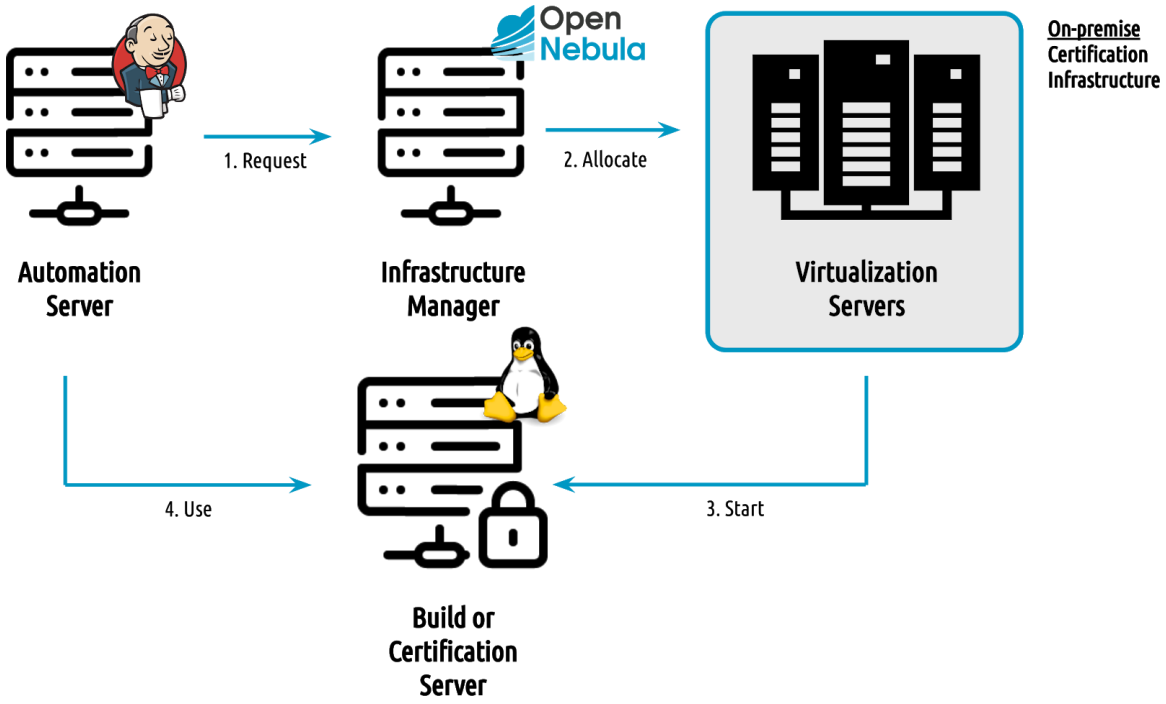


Figure 1.2. Infrastructure resource allocation process

The Automation Server then deploys required dependencies and code for build or certification on allocated build/certification servers and executes the particular process. This approach is also an example of "dogfooding", as the infrastructure uses the recent stable OpenNebula to manage computing resources for build and certification of new EdgeNebula releases.

On-premise infrastructure covers mostly requirements for

- native x86-64 and emulated ARM64 hardware and
- Debian, Ubuntu, CentOS / Red Hat Enterprise Linux operating systems

based types of builds and certifications.

For the builds and certifications which do not fit into on-premise infrastructure (due to resource shortage or unavailability of specific hardware/software), the Infrastructure Manager on-demand deploys the new ephemeral clusters on remote Edge providers and gets build/certification resources from them.

The resource allocation process which utilizes the remote on-demand Edge infrastructure is displayed in Figure 1.3. Automation Server (step 1) requests the resources from the Infrastructure Manager, which does not find suitable virtualization servers to use. It executes the OpenNebula provisioning tools (OneProvision, step 2), the hidden core of the evolving EdgeProvision. Tools contact the remote Edge infrastructure provider (step 3), provision a group of suitable servers and configure them to run KVM hypervisor. Ready to use remote Edge virtualization cluster is automatically added and enabled (step 4) in the Infrastructure Manager. Resource allocation process then continues similarly as with on-premises allocation - starts a group of dedicated Build and Certification Servers (step 5) on new Edge infrastructure, which are passed back to Automation Server (step 6) to follow with appropriate next build or certification steps.

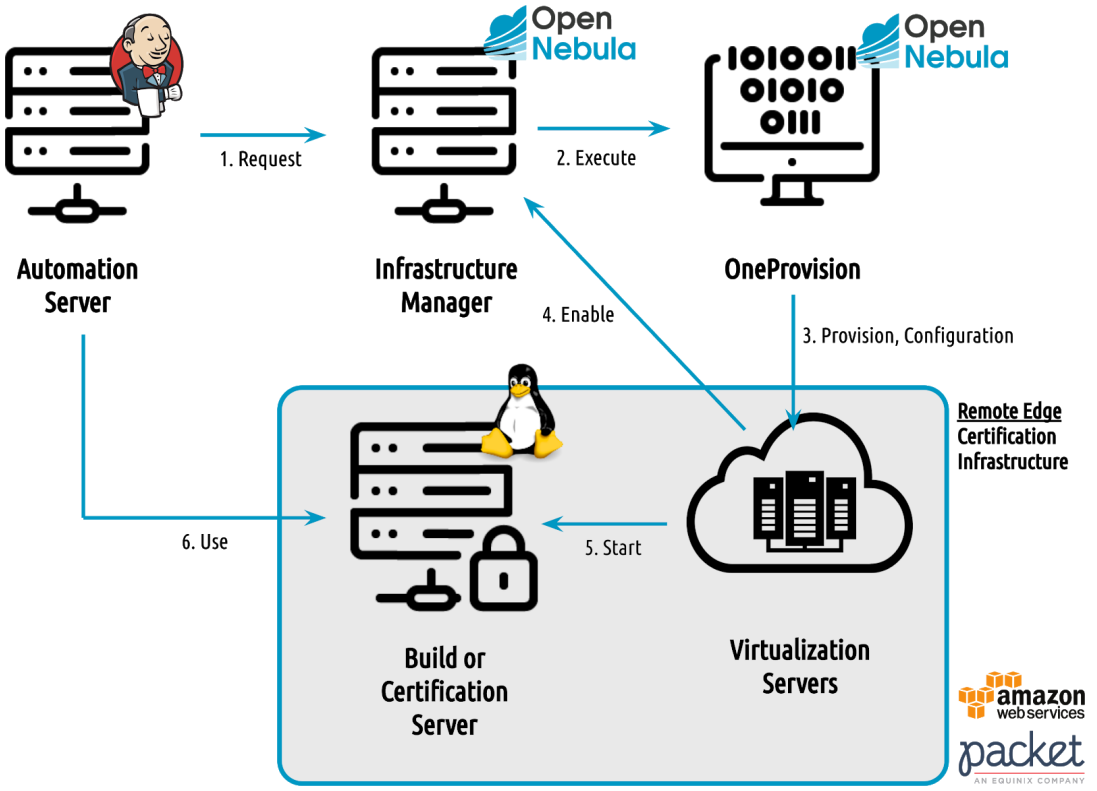


Figure 1.3. Infrastructure resource allocation with on-demand Edge location

On-demand Edge provisioning can benefit from supported providers and their offerings which are compatible with current OpenNebula (5.12) are presented in Table 1.1. This presents a complete list of options, but obviously not all possible combinations (instance type + location) are offered by the providers:

Provider	Host OS	Instance Types	Locations
Amazon EC2[2]	CentOS 7 Ubuntu 16.04 **Ubuntu 18.04**	Up to 20[3]	Up to 23[4]
		E.g. m5dn.metal, u-12tb1.metal, **i3.metal**, m5n.metal, u-6tb1.metal, u-9tb1.metal, u-18tb1.metal, r5dn.metal, r5n.metal, u-24tb1.metal, c5.metal, i3en.metal, r5.metal, r5d.metal, m5.metal, c5d.metal, g4dn.metal, m5d.metal, z1d.metal, c5n.metal	E.g. Africa (Cape Town), Asia-Pacific (Hong Kong, Mumbai, Osaka-Local, Seoul, Singapore, Sydney, Tokyo), Canada (Central), Europe (Frankfurt, Ireland, London, Paris, Stockholm, Milan), Middle East (Bahrain), South America (São Paulo), US East (**N. Virginia**, Ohio), US West (Northern California, Oregon), AWS GovCloud (US-West), AWS GovCloud (US-East)

[2] https://aws.amazon.com/ec2/
[3] https://aws.amazon.com/ec2/instance-types/
[4] https://aws.amazon.com/about-aws/global-infrastructure/regional-product-services/

Packet[5]	CentOS 7 Ubuntu 16.04 **Ubuntu 18.04**	Up to 15[6]	Up to 21[7]
		E.g. **t1.small.x86**, c1.small.x86, c1.xlarge.x86, m1.xlarge.x86, x1.small.x86, s1.large.x86, c3.small.x86, c3.medium.x86, m3.large.x86, s3.xlarge.x86, c2.medium.x86, m2.xlarge.x86, n2.xlarge.x86, x2.xlarge.x86, g2.large.x86	E.g. **Amsterdam**, Ashburn, Dallas, New York City, Silicon Valley, Singapore, Tokyo, Atlanta, Chicago, Detroit, Houston, Kansas City, Los Angeles, Phoenix, Pittsburgh, Seattle, Toronto, Frankfurt, Hong Kong, Marseille, Sydney

Table 1.1. On-demand Edge certification infrastructure configuration options (07/2020)

On-demand Edge infrastructure covers mostly requirements for

- native x86-64 and emulated ARM64 hardware and
- on-going support for native ARM64 hardware
- Debian, Ubuntu, CentOS operating systems
- highly performant or other specific (e.g. high-density CPUs, large memory, GPU, InfiniBand, etc.)

based types of builds and certifications, which does not fit or are not possible in on-premise infrastructure. For the on-demand certification we by default use a single combination from each provider. This infrastructure defaults are highlighted in Table 1.1 (e.g. default for Amazon EC2's is i3.metal instance in US East (N. Virginia) region and Ubuntu 18.04 base operating system).

The overhead to prepare new on-demand Edge infrastructure for the certification process (when compared to the on-premise which is available instantly), depends on the selected provider, instance type, location or potential infrastructure problems on the provider's side. For near distant locations, the initial average overhead in the build/certification servers allocation process is **12 minutes**, i.e.

- **6 minutes** to provision fresh new infrastructure
- **6 minutes** to configure and get ready infrastructure hosts for OpenNebula

for steps 3 and 4 in Figure 1.3. The measured overhead is an average of historic deployment times of new on-demand Edge infrastructure on Packet's Amsterdam (Netherlands) location from the on-premise company infrastructure in Madrid (Spain).

(Note: We publicly demonstrated use of large on-demand Edge infrastructure around the world on Packet with accurate measured times of individual phases and break down by locations in https://opennebula.io/opennebula-a-lightning-fast-video-gaming-edge-use-case-2).

[5] https://www.packet.com
[6] https://www.packet.com/cloud/servers
[7] https://www.packet.com/cloud/locations

Figure 1.4. Locations for on-demand Edge use on Amazon EC2, July 2020
(after https://aws.amazon.com/about-aws/global-infrastructure)

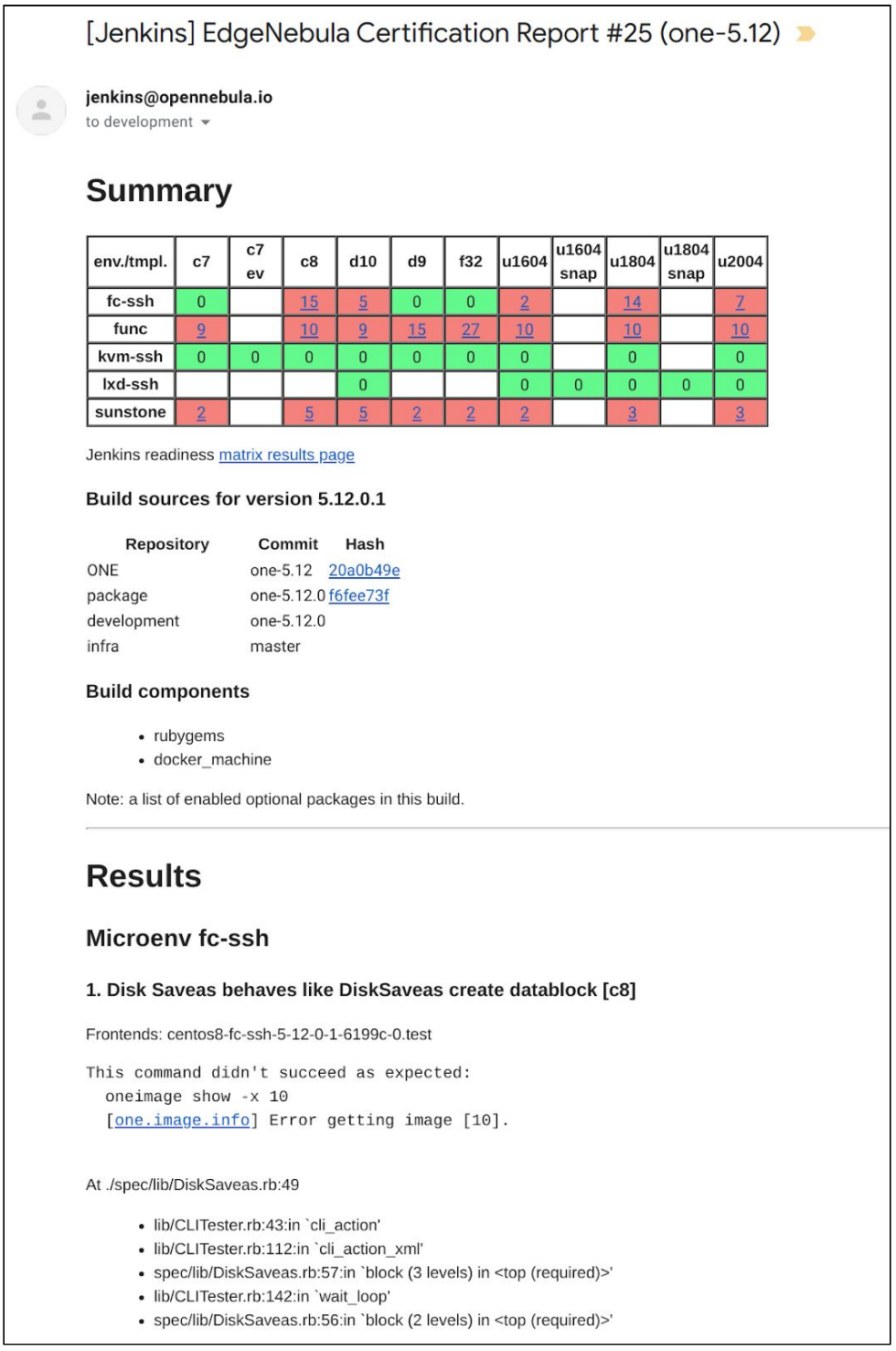
1.3 Certification Process

Runs the test suites on the computing resources provided by Certification Servers for the various verification scenarios and different environments and platforms.

Test suites use Ruby programming language and RSpec testing framework as main (but not a single) means to describe verification scenarios.

It transparently integrates with other languages and their native testing frameworks (e.g. unittest for Python, JUnit for Java and testing for Go), which allows to easily aggregate the results from different parts into a single tool.

After the process ends, a certification report is generated (see Figure 1.5) with an overview of verified platforms and scenarios, followed by detailed description of each failed test.



[Jenkins] EdgeNebula Certification Report #25 (one-5.12)

jenkins@opennebula.io
to development

Summary

env./tmpl.	c7	c7 ev	c8	d10	d9	f32	u1604	u1604 snap	u1804	u1804 snap	u2004
fc-ssh	0		15	5	0	0	2		14		7
func	9		10	9	15	27	10		10		10
kvm-ssh	0	0	0	0	0	0	0		0		0
lxd-ssh				0			0	0	0	0	0
sunstone	2		5	5	2	2	2		3		3

Jenkins readiness matrix results page

Build sources for version 5.12.0.1

Repository	Commit	Hash
ONE	one-5.12	20a0b49e
package	one-5.12.0	f6fee73f
development	one-5.12.0	
infra	master	

Build components

- rubygems
- docker_machine

Note: a list of enabled optional packages in this build.

Results

Microenv fc-ssh

1. Disk Saveas behaves like DiskSaveas create datablock [c8]

Frontends: centos8-fc-ssh-5-12-0-1-6199c-0.test

```
This command didn't succeed as expected:
  oneimage show -x 10
  [one.image.info] Error getting image [10].
```

At ./spec/lib/DiskSaveas.rb:49

- lib/CLITester.rb:43:in `cli_action'
- lib/CLITester.rb:112:in `cli_action_xml'
- spec/lib/DiskSaveas.rb:57:in `block (3 levels) in <top (required)>'
- lib/CLITester.rb:142:in `wait_loop'
- spec/lib/DiskSaveas.rb:56:in `block (2 levels) in <top (required)>'

Figure 1.5. Example of Certification Report

Testing is done by following components

- Operating System (e.g. CentOS 7, 8; Debian 9, 10; Ubuntu 16.04, 18.04, 20.04)
- Product Environment (e.g. KVM SSH, LXD SSH, Firecracker SSH, …)

Certification is not running on the granularity of each verification scenario. Group of scenarios and preconfigured state of EdgeNebula with dependencies (e.g. different storage solutions) forms a testing Product Environment. Combination of Product Environments and Operating Systems make the sparse testing matrix, which affects what and where will be tested during the certification.

1.4 Download Servers

Host the released product as a downloadable archive and distribution (rpm/deb) repository, they are common Linux (virtual) machines with large storage and running a web server. On successful Certification Process and after approval by the product owner, the Automation Server uploads the build artifacts to the Download Servers and makes them available to the users.

2. Software Requirements Verification

This section includes a detailed list of the extensions implemented to verify the functionality of the software developed in ONEedge. Tests and extensions of the verification framework are detailed for each component and grouped for each software requirement implemented during the cycle. For each requirement we include a summary of the extensions performed in the testing and certification infrastructure. Also, for each requirement, we list the verification scenarios that have been addressed and a description of the functionality tested to fulfill the proposed scenarios.

2.1 Edge Instance Manager (CPNT1)

SR1.2. Automatic Product Upgrade
Status: IN PROGRESS
Description: Testing has been extended to cover new tools to upgrade configuration of EdgeStack components. Verification focuses both on testing the tools in isolated controlled environments and on real use in the testing environments which upgrade real components and their configurations.
Verification Scenarios: • [VS1.2.1] Tests configuration upgrade of the EdgeStack components among various version releases. Verifies that content of upgraded configurations corresponds to steps done during upgrade.

2.2 Edge Workload Orchestration and Management (CPNT2)

SR2.1. Integration with Serverless Hypervisor
Status: DONE
Description: The testing infrastructure has been extended to automatically deploy Firecracker environments on relevant OS distributions, including CentOS 7 and 8, Debian 9 and 10; and Ubuntu 16.04, 18.10 and 20.04. Specific tests have been developed to test the functionality of ONEedge managing Firecracker VMs on relevant storage and networking setups.
Verification Scenarios: • [VS2.1.1] Tests that check that Firecracker nodes can be added to ONEedge, and verify that monitoring information and state are properly obtained from the nodes.

- [VS2.1.2] Tests that deploy Firecracker micro-VM and perform basic operations (create, terminate, power off and attach/detach operations). The state of the VMs and the operations performed are verified on the nodes and front-end.
- [VS2.1.3] Tests that deploy a Firecracker VM and check (ping and SSH) its network configuration and connectivity using different network drivers (bridge, VLAN and VXLAN).
- [VS2.1.4] Tests that download sample docker images from Docker Hub and prepare them automatically to be deployed in ONEedge. The tests verify that the contextualization process has been correctly executed (SSH public key installation and network configuration).

SR2.3. Secure and Scalable Distributed Monitoring

Status: DONE

Description: The test suite has been adapted and extended to verify the new monitoring system. Also we have developed new scalability tests that stress the new monitor components to assess the response time and behavior of the system.

Verification Scenarios:

- [VS2.3.1] Tests that verify that probes send monitor data encrypted with a pre-defined key.
- [VS2.3.2] Tests send probe information at different frequencies.
- [VS2.3.3] Tests stress monitor system and API server. We measure the number of API requests served per second on different load conditions. Also, tests evaluate the monitor probes to assess their scalability with a high number of VMs/containers per node.

SR2.5. Integration with Remote VMware vCenter Service

Status: DONE

Description: We have installed NSX on the existing VMware infrastructure. Specific testing networks have been created in the testing environment to verify different use-cases. Also the network test suite for vCenter has been extended to include the new features.

Verification Scenarios:

- [VS2.5.1] Tests to define security groups and verify that security groups are created and associated to the corresponding resources. Security group creation and management is verified for NSX-t and NSX-v.
- [VS2.5.2] Tests that associate VMs to existing security groups and verify that the traffic is properly filtered.

SR2.6. VNF Support
Status: DONE
Description: We have developed new tests to verify the NUMA-aware placement policies. These tests verifies the code at a functional logical level (i.e. placement policies, capacity checks) as well as the hypervisor interaction (i.e. the VM actually has the selected topology).
Verification Scenarios: • [VS2.6.1] Tests to check that hosts correctly report their NUMA topology and available capacity. • [VS2.6.2] Tests to check the different allocation policies implemented by the scheduler: core, thread and shared. • [VS2.6.3] Tests to verify that the VM is configured and allocated in the host according to the selected policy.

SR2.8. Complete Service Flows
Status: IN PROGRESS
Description: We have developed new tests to verify OneFlow's new network capabilities.
Verification Scenarios: • [VS2.7.3] Tests that define a flow with an associated network. The test verifies that the network is created and, once the flow ends, the network is also terminated.

SR2.9. Web UI Extensions
Status: IN PROGRESS
Description: We have developed new tests using the selenium framework to test the extensions developed in each SR.
Verification Scenarios: • Tests that verify the functionality of the new dialogs and widgets.

2.3 Edge Provider Selection (CPNT3)

SR3.4. Driver Maintenance Process
Status: IN PROGRESS
Description: Two drivers are now available (Amazon EC2 and Packet) which lays the foundation to define a maintenance process. Testing has been implemented to exercise these provision drivers
Verification Scenarios: • [VS3.4.3] Tests that deploy a remote provision (one host in one cluster on an edge location, adds a new public virtual network, exports an Alpine VM and checks that all the elements are correctly created and configured. Also creates a VM with private networking, do a NIC hotplug and detach. terminate s the VM. Creates a VM with public and private networking. Deletes the remote provision and all associated resources and checks that the removal is successful within OpenNebula.

2.4 Edge Infrastructure Provision and Deployment (CPNT4)

SR4.1. Reliable Edge Resource Provision
Status: IN PROGRESS
Description: Edge provision testing was enhanced to leverage multi-staged error handling with a new prototype of orphaned deployments cleaner.
Verification Scenarios: • [VS4.1.1] Tests do provision of new resources on Edge location. The provisioning process is running with advanced error handling to retry and recover from the error situations. It fails only during consistent non-recoverable errors and reverts changes done so far. On success, the functionality of deployed locations is validated by running test virtual machines and operational tests. After tests end, Edge location is unprovisioned and resources released. • [VS4.1.2] Common Edge location provision testing with extra mechanism triggered at the very end to detect still allocated resources. These resources are forcefully released back to the provider.

SR4.2 Usability, Functionality and Scalability of Provision
Status: IN PROGRESS

Description: Testing scenarios were extended to cover testing of new objects supported by provisioning tools.

Verification Scenarios:

- [VS4.2.3] Tests deploy dummy locations with various end-user virtual objects (e.g. virtual machine templates, virtual network templates, service templates, images, marketplace images) and validate the particular objects were created.

SR4.3. Provision Template for Reference Architectures

Status: IN PROGRESS

Description: Testing extended for introduced complete cluster reference templates.

Verification Scenarios:

- [VS4.3.1] Tests create a provision from complete cluster reference templates to supported Edge providers.

2.5 Edge Apps Marketplace (CPNT5)

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

Status: IN PROGRESS

Description: The Kubernetes appliance allows for single master deployment within OpenNebula. It can also be deployed as a OneFlow service, which lays off the foundation for a elastic Kubernetes cluster using OneFlow elasticity rules.

Verification Scenarios:

- [VS5.2.1] Deploy the OneFlow service and check if the needed VMs are created successfully. Check the contextualization process correctly configured in the Kubernetes cluster.

SR5.3. Integration with Application Containers Marketplace

Status: IN PROGRESS

Description: New marketplace offering all the Docker Hub applications have been added to OpenNebula 5.12.

Verification Scenarios:

- [VS5.3.1] Test to export a particular Docker appliance from Docker Hub to an OpenNebula datastore. Instantiate a VM based on the exported Docker appliance. Check connectivity through SSH. Terminate the VM and remove the image entry and template from OpenNebula. Done for the following Docker appliances: Alpine, Ubuntu, CentOS, Debian.

SR5.4. New Edge Applications Marketplace Entries

Status: DONE

Description: Kubernetes appliance in the OpenNebula marketplace has been updated (including minor enhancements) to version 1.18.3.

Verification Scenarios:

- [VS5.4.1] This functionality is tested in the coverage of [VS5.1.2] and [VS5.2.1].