D5.10 COGNIT Framework - Demo - a

Version 1.0

30 April 2024

Abstract

COGNIT is an AI-Enabled Adaptive Serverless Framework for the Cognitive Cloud-Edge Continuum that enables the seamless, transparent, and trustworthy integration of data processing resources from providers and on-premises data centers in the cloud-edge continuum, and their automatic and intelligent adaptation to optimise where and how data is processed according to application requirements, changes in application demands and behaviour, and the operation of the infrastructure in terms of the main environmental sustainability metrics. This document provides both a demonstration of how to deploy the COGNIT Framework on a target infrastructure, and a demonstration of some of the capabilities of the COGNIT Framework using the COGNIT testbed hosted by RISE.

This project is funded by the European Union's Horizon Europe research and innovation programme under Grant Agreement 101092711 – SovereignEdge.Cognit

Deliverable Metadata

Project Title:	A Cognitive Serverless Framework for the Cloud-Edge Continuum
Project Acronym:	SovereignEdge.Cognit
Call:	HORIZON-CL4-2022-DATA-01-02
Grant Agreement:	101092711
WP number and Title:	WP5. Adaptive Serverless Framework Integration and Validation
Nature:	DEM: Demonstrator, Pilot, Prototype
Dissemination Level:	PU: Public
Version:	1.0
Contractual Date of Delivery:	31/03/2024
Actual Date of Delivery:	30/04/2024
Lead Author:	Thomas Ohlson Timoudas (RISE)
Authors:	Antonio Álvarez (OpenNebula), Monowar Bhuyan (UMU), Simon Bonér (UMU), Aritz Brosa (Ikerlan), Daniel Clavijo (OpenNebula), Johan Kristiansson (RISE), Marco Mancini (OpenNebula), Alberto P. Martí (OpenNebula), Goiuri Peralta (Ikerlan), Constantino Vázquez (OpenNebula), Pavel Czerny (OpenNebula).
Status:	Submitted

Document History

Version	Issue Date	Status[1]	Content and changes
0.1	25/04/2024	Draft	Initial Draft
0.2	29/04/2024	Peer-Reviewed	Reviewed Draft
1.0	30/04/2024	Submitted	Final Version

Peer Review History

Version	Peer Review Date	Reviewed By
0.1	29/04/2024	Nikolaos Matskanis (CETIC)
0.1	29/04/2024	Antonio Álvarez (OpenNebula)

Summary of Changes from Previous Versions

First Version of Deliverable D5.10

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

Executive Summary

Deliverable D5.10 presents the demonstrations of the first release of the COGNIT Framework, issued in M15. It includes both a demonstration of how to deploy the COGNIT Framework on a target infrastructure, and a demonstration of the COGNIT Framework in an operational environment using the COGNIT testbed at RISE.

The first demonstration shows how to deploy the complete COGNIT Framework on a target infrastructure using the **OpsForge** tool, which automatically integrates and deploys the entire COGNIT software stack. Specifically, it sets up the following COGNIT components: 1) The Cloud-Edge Manager, 2) the AI-Enabled Orchestrator, 3) the Serverless Runtime, and 4) the Provisioning Engine. As an example, this demonstration deploys the COGNIT Framework on public cloud resources from AWS.

The second demonstration uses the COGNIT testbed to show some of the capabilities of the COGNIT Framework, considering three different scenarios:

1) A device requests a Serverless Runtime and offloads a function.
2) A device updates its requirements, which triggers a migration of the Serverless Runtime.
3) Serverless Runtimes have to be migrated automatically due to changes in the underlying cloud-edge infrastructure.

Apart from this report, the software integration process and infrastructure is further detailed, together with the verification of the software requirements, in Deliverable D5.3, and the public repositories containing the open source code for automating this integration and deployment process (plus its associated documentation), as well as the a number of toolkits developed for the Use Cases, are presented in Deliverable D5.7.

This deliverable has been released at the end of the Second Research & Innovation Cycle (M15), and will be updated with incremental releases in M27 and M33.

Table of Contents

Abbreviations and Acronyms

AI	Artificial Intelligence
AWS	Amazon Web Services
EC2	(Amazon) Elastic Compute Cloud
FaaS	Function as a Service
IP	Internet Protocol
VM	Virtual Machine
VPC	Virtual Private Cloud
YAML	Yaml Ain't a markup language

1. Introduction

The initial version of the COGNIT Framework Demo (Deliverable D5.10), released in M15, includes both a demonstration of how to deploy the COGNIT Framework on a target infrastructure, and a demonstration of the COGNIT Framework in an operational environment using the COGNIT testbed.

Section 2 contains the details of a first demonstration showing how to deploy the complete COGNIT Framework on a target infrastructure using the COGNIT OpsForge tool, which automatically integrates and deploys the entire COGNIT software stack. As an example, this demonstration deploys the COGNIT Framework on AWS.

Section 3 contains the details of a second demonstration that leverages the existing COGNIT testbed hosted by RISE to show some of the capabilities of the COGNIT Framework in an operational environment. This demo covers three different scenarios:

1) A device requests a Serverless Runtime and offloads a function.
2) A device updates its requirements, which triggers a migration of the Serverless Runtime.
3) Serverless Runtimes have to be migrated automatically due to changes in the underlying cloud-edge infrastructure.

The document ends with conclusions in Section 4.

2. OpsForge Demo

The new component **cognit-opsforge** allows to easily deploy the complete COGNIT Framework on a target infrastructure (like an on-premise data centre or a public cloud) and turn it into a Cognitive Serverless Framework for the Cloud-Edge Continuum.

This demo will show how to deploy the COGNIT Framework on the AWS Public Cloud using OpsForge. The M15 release of OpsForge can automatically deploy and configure the following components on the target infrastructure:

- Cloud-Edge Manager (i.e. based on OpenNebula) with the Serverless Template.
- Provisioning Engine.
- AI-Enabled Orchestrator (only the initial configuration).

Currently, OpsForge will only create the needed virtual resources to contain the AI-Enabled Orchestrator. During the demo we will demonstrate how to manually deploy the AI-Enabled Orchestrator. Later versions of the OpsForge Tool will include the AI-Enabled Orchestrator component in the fully automated setup.

2.1 OpsForge Configuration

In order to install the OpsForge, the user must first install the runtime dependencies needed by the tool. The user must then clone the git repository of COGNIT OpsForge specifying the proper version, which in this case would be release-cognit-1.0. Using the command line, the user should execute the commands below, which will create a copy of COGNIT OpsForge in the currently active directory.

```
Unset
git clone https://github.com/SovereignEdgeEU-COGNIT/cognit-ops-forge.git
—recursive
git checkout release-cognit-1.0
```

In order to run the tool, we need first to create a YAML file (for this demonstration, we call it *aws.yaml*) that contains different configuration options for the deployment, and will be used as input for the OpsForge tool. Since we are targeting a deployment on the AWS public cloud, this configuration file with contain options related to AWS, as below:

```
Unset
:infra:
 :aws:
   :ec2_instance_type: t2.medium
   :volume_size: 125
   :region: "us-east-1"
   :ssh_key: "dann1"
```

```
    :ssh_key_path: '~/.ssh/id_rsa'
:cognit:
 :engine:
   :port: 1337
   :version: release-cognit-1.0
 :ai_orchestrator:
   :version: release-cognit-1.0
 :cloud:
   :version: 6.8
   :ee_token: "XXXX:XXXX"
   :web_ports:
     :main: 80
     :next_gen: 443
   :extensions:
     :version: release-cognit-1.0
```

All the possible parameters in the configuration file are shown in the template above. The only mandatory parameters to set values for are:

1. the :**region** and :**ssh_key** in :**aws** in :**infra**, which needs to specify credentials and characteristics of the desired target infrastructure in AWS, and
2. the :**ee_token** in :**cloud** in :**cognit**, that needs to contain a valid OpenNebula token to access the enterprise edition repositories.

For the other (optional) parameters, their default values will be used if they have no values set. The default value for each parameter is shown in the verbose template above.

2.2 COGNIT Framework Provisioning

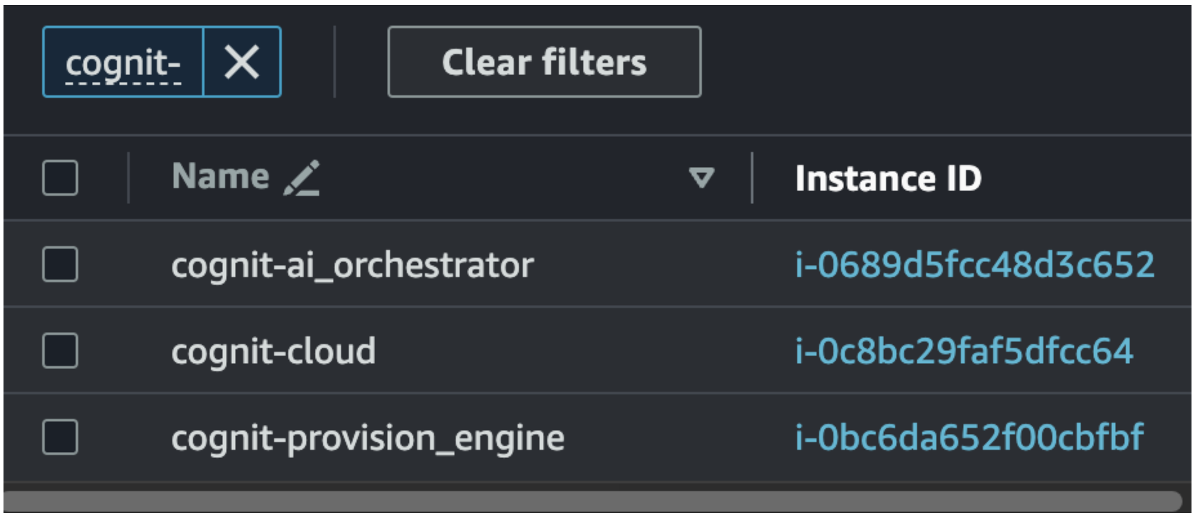
Once the configuration file has been properly set up, we can deploy the COGNIT Framework using the following command within the folder of the cloned github repository (see Section 2.1 above).

```
Unset
./opsforge deploy aws.yaml
```

The OpsForge tool will perform several tasks that are described in the following. First it provisions the infrastructure on AWS as it is shown in the log reported below:

```
Unset
Setting up infrastructure on AWS
Infrastructure on AWS has been deployed
Took 63.615924 seconds
```

Opsforge creates three EC2 instances, a Virtual Private Cloud (VPC) and the networking configuration required by the COGNIT components to communicate with each other provisioned in the specified region as shown in the following image:



Name	Instance ID
cognit-ai_orchestrator	i-0689d5fcc48d3c652
cognit-cloud	i-0c8bc29faf5dfcc64
cognit-provision_engine	i-0bc6da652f00cbfbf

Figure 2.1: The provisioned AWS EC2 instances and the corresponding COGNIT components hosted there.

After that, it configures those EC2 instances with the packages required by each component using the corresponding github repositories as shown in the log below:

```
Unset
Installing Cloud-Edge Manager, Provisioning Engine and AI-Enabled
Orchestrator
Frontend and Provisioning Engine installed
Took 439.582241 seconds
```

```
Unset
ubuntu@ip-10-0-1-93:~$ oned --version | head -n 1
OpenNebula 6.8.2 (7e331ab4) Enterprise Edition
curl http://"ec2-3-93-35-232.compute-1.amazonaws.com":6969/server/version
"1.3.3"⏎
```

Once the packages have been installed, the Cloud-Edge Manager is updated with Serverless Runtimes Appliances[2] for each of the use cases. At the moment, the source of these appliances is a placeholder as the procedure to automate this part has not yet been implemented.

[2] https://docs.opennebula.io/6.8/marketplace/appliances/index.html

```
Unset
Setting up Cloud-Edge Manager for Cognit
Frontend ready for Cognit
Took 28.971458 seconds
```

If everything goes well, the output will contain information about the infrastructure.

```
Unset
Infrastructure
{
  "cloud": "ec2-3-236-130-204.compute-1.amazonaws.com",
  "engine": "ec2-3-93-35-232.compute-1.amazonaws.com",
  "ai_orchestrator": "ec2-3-224-211-145.compute-1.amazonaws.com"
}

Connect to these hosts with the <ubuntu> user using the provided ssh key'
Logs available at ./opsforge.log'

Take a look at AWS cluster provisioning in order to setup your KVM cluster
https://docs.opennebula.org/stable/provision_clusters/providers/aws_provider
.html#aws-provider

After that, take a look at the Energy Consumption extension
https://github.com/SovereignEdgeEU-COGNIT/opennebula-extensions?tab=readme-o
v-file#scaphandre-extension
```

Once the deployment is completed, a default user (*oneadmin*) is available for connecting and using the COGNIT Framework. Furthermore, we can proceed to install the AI-Enabled Orchestrator that at the moment has not yet been fully integrated into the OpsForge tool.

To manually deploy the AI-Enabled Orchestrator, the user first needs to connect with SSH to the EC2 instance set up by OpsForge for hosting to the AI-Enabled Orchestrator (Instance-ID i-689d5fcc48d3c652 in Figure 2.1) and create the following configuration file with environment variables:

```
Unset
export LANG=en_US.UTF-8
export LANGUAGE=en_US.UTF-8
export LC_ALL=en_US.UTF-8
export LC_CTYPE=UTF-8
export TZ=Europe/Stockholm
export ENVSERVER_VERBOSE="false"  # Verbose logging
export ENVSERVER_HOST="addr"     # Address to the Database manager/envserver
export ENVSERVER_PORT="50080"      # Port for the Database manager/envserver
export ENVSERVER_TLS="false"        # TLS for the Database manager/envserver
export ENVSERVER_DB_HOST="addr"     # Address to the Timescale DB
export ENVSERVER_DB_USER="postgres" # User credentials for the Timescale DB
export ENVSERVER_DB_PORT="5432"     # Port for the Timescale DB
export ENVSERVER_DB_PASSWORD="pass" # User credentials for the Timescale DB
export ONED_PASS="pass"             # User credentials for OneD
export ONED_ADDR="addr"             # Address to OneD
export ML_PORT="50090"              # Port for the MLServer
export ML_HOST="mlserver"           # Address to MLServer
export ML_INSECURE="false"          # TLS to MLServer
```

We need to clone the AI-Enabled Orchestrator github repository https://github.com/SovereignEdgeEU-COGNIT/ai-orchestrator-env in the VM and then execute the docker compose command in the folder of the repository:

```
Unset
source .env
docker compose --file=./docker-compose-one.yml up
```

2.3 Deprovisioning of the COGNIT Framework

In order to undeploy the COGNIT Framework and free the resources, the following command can be used, which again should be executed in the directory where the configuration file (in this case *aws.yaml*) is located:

```
Unset
./opsforge clean
Destroying resources infrastructure
COGNIT deployment successfully destroyed
```

In the following section we will show a demo on how to use the device client with the COGNIT Framework.

3. COGNIT Framework DEMO

The COGNIT Framework demo is performed using the COGNIT Testbed that has two primary hosts (*p02r11srv01* and *p02r11srv15*) within one cluster.

In order to perform some simulation scenarios, the host *p02r11srv01* has been set with the label "*ENERGY_RENEWABLE=YES*", to simulate that the host is currently running on renewable energy. The purpose is to simulate how the COGNIT Framework handles the case when a device would like to use computational resources on servers powered mainly by renewable energy.

In order to offload functions on hosts with the renewable energy label set to "YES", the device must set the energy scheduling policy to be renewable in the configuration of the Serverless Runtime.

In this demo, we consider three different scenarios for showing some capabilities of the COGNIT Framework as follows:

1) Requesting a Serverless Runtime and offloading a function
2) Update requirements from the device triggering Serverless Runtime migration
3) Migration of Serverless Runtimes due to changes in the infrastructure

3.1 Requesting a Serverless Runtime and Offloading a function

This subsection demonstrates the process of requesting a Serverless Runtime and offloading a function from the COGNIT Device Client (Python version) to it.

First of all the device shall create a configuration file (e.g. *cognit.yml*) with the endpoint and credentials for the COGNIT Framework deployment as below:

```Python
endpoint: "cognit-pe" # IP of your Provisioning Engine
port: 1337 # Port of the Provisioning Engine
pe_usr: "oneadmin" # Provisioning Engine Username.
pe_pwd: "oneadmin_password" # Provisioning Engine Password.
sr_port: 8000 # Serverless Runtime port
```

In this case, the endpoint was configured to point to an instance in the COGNIT Testbed, but in the example deployment demonstrated in Section 2, the endpoint parameter would have been set to "ec2-3-93-35-232.compute-1.amazonaws.com". The parameter sr_port sets the port for the Serverless Runtime that will be created. The IP address of the Serverless Runtime will be returned to the device runtime once created.

To create a Serverless Runtime and offload a function using python code, first we need to import the *cognit* module:

```python
from cognit import (
    ServerlessRuntimeConfig,
    ServerlessRuntimeContext,
    FaaSState
)
```

For demonstration purposes, let’s define the following simple function to be offloaded to the Serverless Runtime:

```python
def sum(a: int, b: int):
    return a + b
```

In order to request a Serverless Runtime, a set of requirements must be instantiated and passed to the Provisioning Engine. In the following, the main requirement is the name of the flavour containing the application dependencies and libraries needed to execute the function. Additionally, the label “ENERGY_RENEWABLE” in the requirements, can be configured through the *EnergySchedulingPolicy*. Currently, values below 50 result in setting “ENERGY_RENEWABLE=NO” in the requirements while values above 50 result in “ENERGY_RENEWABLE=YES”. In this case, we set a requirement for “ENERGY_RENEWABLE=NO”, by setting the value of *EnergySchedulingPolicy* to 30.

```python
sr_conf = ServerlessRuntimeConfig()
sr_conf.name = "Use Case Energy"
sr_conf.faas_flavour = "Energy"
sr_conf.scheduling_policies = [EnergySchedulingPolicy(30)]
```

Once the requirements have been set, the device can proceed to create a Serverless Runtime using the configuration file created previously:

```python
cognit_sr = ServerlessRuntimeContext(config_path="cognit.yml")
cognit_sr.create(sr_conf)
```

The AI-Enabled Orchestrator will create and place the Serverless Runtime according to the requirements set by the Device Client:

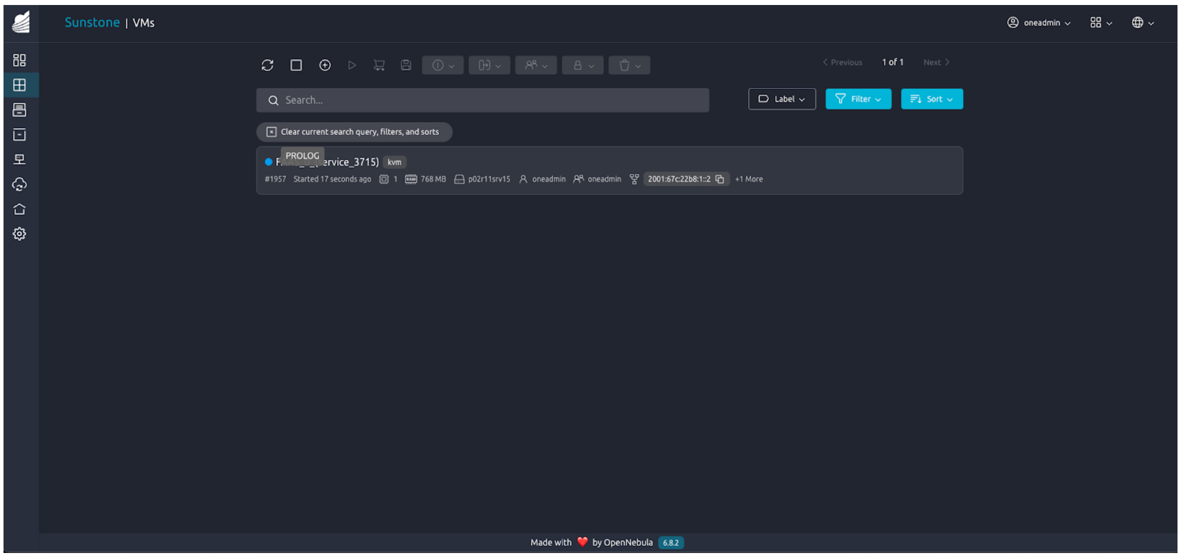
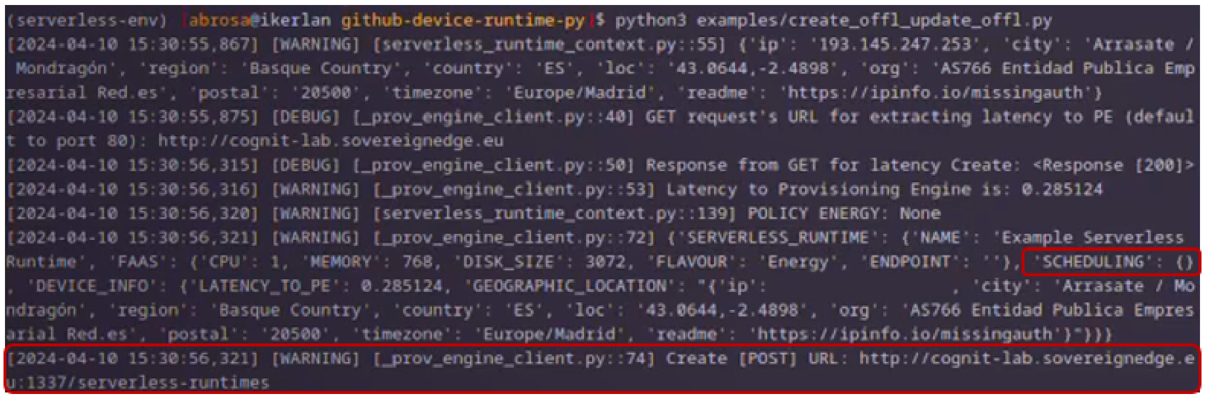


Figure 3.1: A Serverless Runtime has been just created on host p02r11srv15

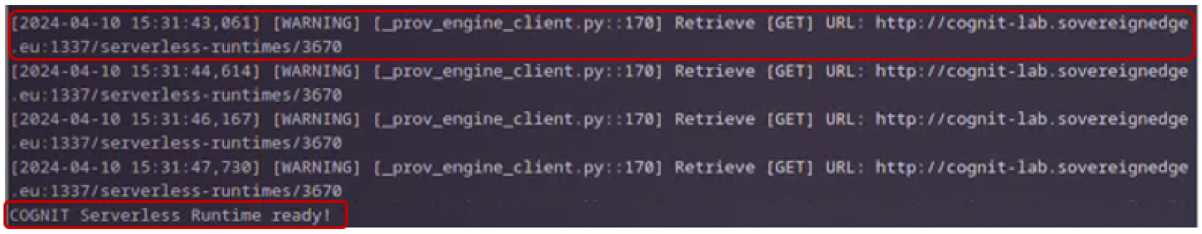


```
(serverless-env) [abrosa@ikerlan github-device-runtime-py]$ python3 examples/create_offl_update_offl.py
[2024-04-10 15:30:55,867] [WARNING] [serverless_runtime_context.py::55] {'ip': '193.145.247.253', 'city': 'Arrasate /
 Mondragón', 'region': 'Basque Country', 'country': 'ES', 'loc': '43.0644,-2.4898', 'org': 'AS766 Entidad Publica Emp
resarial Red.es', 'postal': '20500', 'timezone': 'Europe/Madrid', 'readme': 'https://ipinfo.io/missingauth'}
[2024-04-10 15:30:55,875] [DEBUG] [_prov_engine_client.py::40] GET request's URL for extracting latency to PE (defaul
t to port 80): http://cognit-lab.sovereignedge.eu
[2024-04-10 15:30:56,315] [DEBUG] [_prov_engine_client.py::50] Response from GET for latency Create: <Response [200]>
[2024-04-10 15:30:56,316] [WARNING] [_prov_engine_client.py::53] Latency to Provisioning Engine is: 0.285124
[2024-04-10 15:30:56,320] [WARNING] [serverless_runtime_context.py::139] POLICY ENERGY: None
[2024-04-10 15:30:56,321] [WARNING] [_prov_engine_client.py::72] {'SERVERLESS_RUNTIME': {'NAME': 'Example Serverless
Runtime', 'FAAS': {'CPU': 1, 'MEMORY': 768, 'DISK_SIZE': 3072, 'FLAVOUR': 'Energy', 'ENDPOINT': ''}, 'SCHEDULING': {}
, 'DEVICE_INFO': {'LATENCY_TO_PE': 0.285124, 'GEOGRAPHIC_LOCATION': "{'ip':                  , 'city': 'Arrasate / Mo
ndragón', 'region': 'Basque Country', 'country': 'ES', 'loc': '43.0644,-2.4898', 'org': 'AS766 Entidad Publica Empres
arial Red.es', 'postal': '20500', 'timezone': 'Europe/Madrid', 'readme': 'https://ipinfo.io/missingauth'}"}}}
[2024-04-10 15:30:56,321] [WARNING] [_prov_engine_client.py::74] Create [POST] URL: http://cognit-lab.sovereignedge.e
u:1337/serverless-runtimes
```

Figure 3.2: Created Serverless Runtime info returned to the Device Runtime

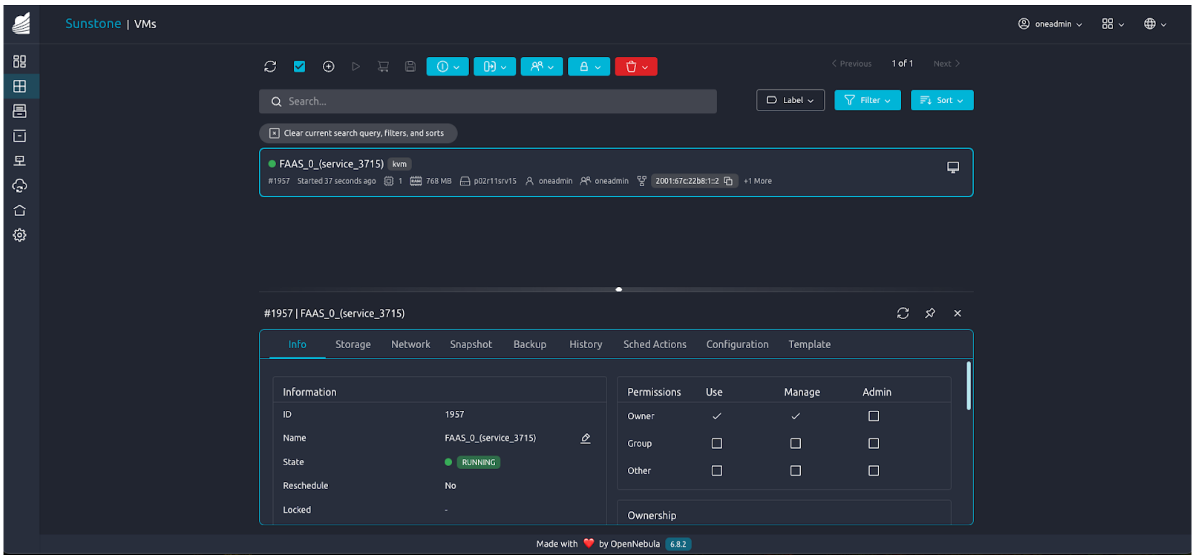
Since it will take time to create a new Serverless Runtime, the device will need to wait, regularly polling the Provisioning Engine, until the Serverless Runtime is ready:

```Python
while cognit_sr.status != FaaSState.RUNNING:

    time.sleep(1)
```



```
[2024-04-10 15:31:43,061] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
[2024-04-10 15:31:44,614] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
[2024-04-10 15:31:46,167] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
[2024-04-10 15:31:47,730] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
COGNIT Serverless Runtime ready!
```

Figure 3.3: Serverless Runtime is running and ready (Device Runtime perspective)



FAAS_0_(service_3715) kvm
#1957 Started 37 seconds ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::2 +1 More

#1957 | FAAS_0_(service_3715)

Information	
ID	1957
Name	FAAS_0_(service_3715)
State	RUNNING
Reschedule	No
Locked	-

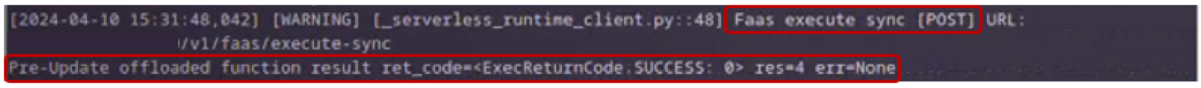
Permissions	Use	Manage	Admin
Owner	✓	✓	☐
Group	☐	☐	☐
Other	☐	☐	☐

Ownership

Figure 3.4: Serverless Runtime is running and ready (Cloud-Edge Manager perspective)

Once the Serverless Runtime is READY, the client can offload the execution of the function using a synchronous call:

```python
result = cognit_sr.call_sync(sum, 2, 2)
```



```
[2024-04-10 15:31:48,042] [WARNING] [_serverless_runtime_client.py::48] Faas execute sync [POST] URL:
                /v1/faas/execute-sync
Pre-Update offloaded function result ret_code=<ExecReturnCode.SUCCESS: 0> res=4 err=None
```

Figure 3.5: Result of the offloaded function (Device Client's output in a terminal)

3.2 Updating Requirements from the Device and Serverless Runtime Migration

It is possible for the device to update the requirements after the Serverless Runtime has been created. After an update, the AI-Enabled Orchestrator can migrate the Serverless Runtime according to the new requirements.

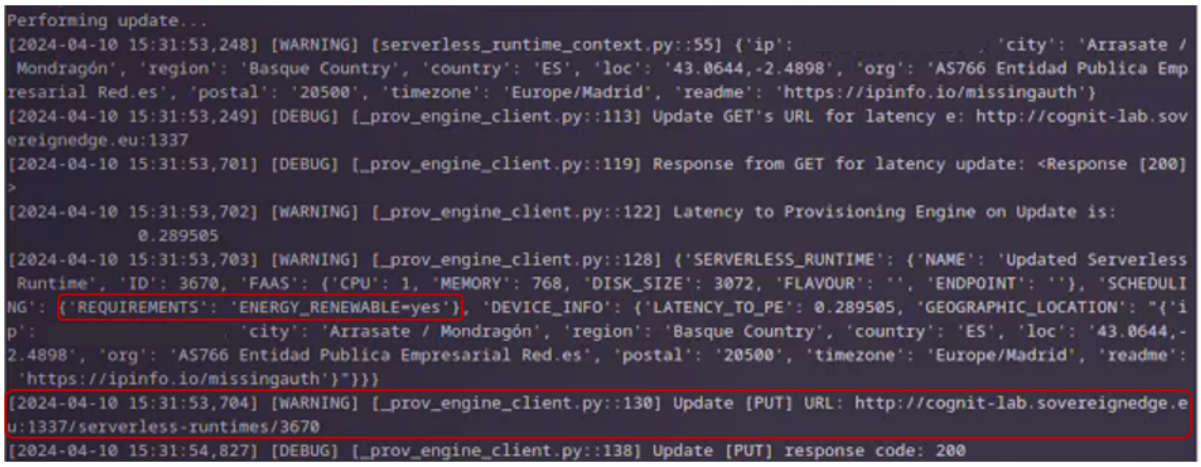
To demonstrate this mechanism, let's update the requirements requesting a host with renewable energy by setting the *EnergySchedulingPolicy* to a value greater than 50 as in the following code:

```python
sr_conf = ServerlessRuntimeConfig()
sr_conf.scheduling_policies = [EnergySchedulingPolicy(80)]
```

To update the requirements for the Serverless Runtime, the client then issues an update to the Provisioning Engine:

```python
cognit_sr.update(sr_conf)
```

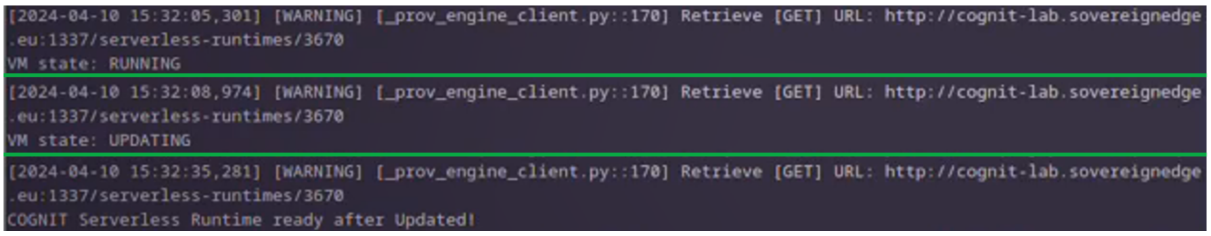
Since the energy scheduling policy has been modified from a 30% to a 80%, the "ENERGY_RENEWABLE" label will be set to "YES" indicating the need for a server powered by renewable energy, as shown below:



```
Performing update...
[2024-04-10 15:31:53,248] [WARNING] [serverless_runtime_context.py::55] {'ip': [redacted], 'city': 'Arrasate /
 Mondragón', 'region': 'Basque Country', 'country': 'ES', 'loc': '43.0644,-2.4898', 'org': 'AS766 Entidad Publica Emp
resarial Red.es', 'postal': '20500', 'timezone': 'Europe/Madrid', 'readme': 'https://ipinfo.io/missingauth'}
[2024-04-10 15:31:53,249] [DEBUG] [_prov_engine_client.py::113] Update GET's URL for latency e: http://cognit-lab.sov
ereignedge.eu:1337
[2024-04-10 15:31:53,701] [DEBUG] [_prov_engine_client.py::119] Response from GET for latency update: <Response [200]
>
[2024-04-10 15:31:53,702] [WARNING] [_prov_engine_client.py::122] Latency to Provisioning Engine on Update is:
            0.289505
[2024-04-10 15:31:53,703] [WARNING] [_prov_engine_client.py::128] {'SERVERLESS_RUNTIME': {'NAME': 'Updated Serverless
 Runtime', 'ID': 3670, 'FAAS': {'CPU': 1, 'MEMORY': 768, 'DISK_SIZE': 3072, 'FLAVOUR': '', 'ENDPOINT': ''}, 'SCHEDULI
NG': {'REQUIREMENTS': 'ENERGY_RENEWABLE=yes'}, 'DEVICE_INFO': {'LATENCY_TO_PE': 0.289505, 'GEOGRAPHIC_LOCATION': "{'i
p': [redacted] 'city': 'Arrasate / Mondragón', 'region': 'Basque Country', 'country': 'ES', 'loc': '43.0644,-
2.4898', 'org': 'AS766 Entidad Publica Empresarial Red.es', 'postal': '20500', 'timezone': 'Europe/Madrid', 'readme':
 'https://ipinfo.io/missingauth'}"}}}
[2024-04-10 15:31:53,704] [WARNING] [_prov_engine_client.py::130] Update [PUT] URL: http://cognit-lab.sovereignedge.e
u:1337/serverless-runtimes/3670
[2024-04-10 15:31:54,827] [DEBUG] [_prov_engine_client.py::138] Update [PUT] response code: 200
```

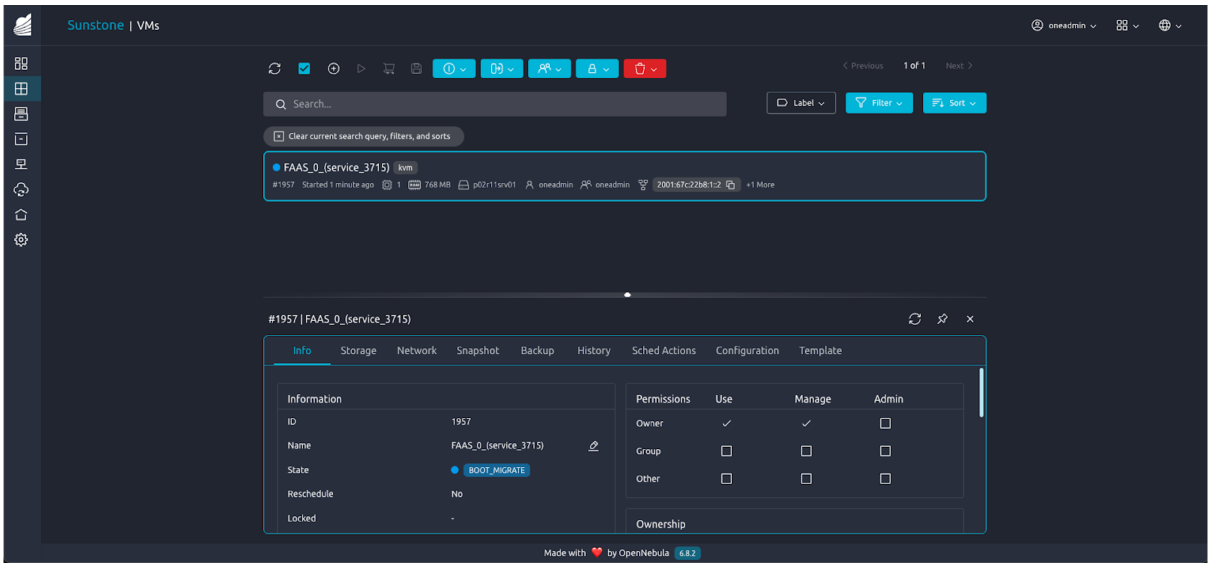
Figure 3.6: Serverless Runtime update requested by the Device Client

This triggers the rescheduling of the Serverless Runtime by the AI-Enabled Orchestrator that based on the new requirements can migrate the Serverless Runtime. During the migration process, the Serverless Runtime will change from a RUNNING state to other intermediate states:



```
[2024-04-10 15:32:05,301] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
VM state: RUNNING
[2024-04-10 15:32:08,974] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
VM state: UPDATING
[2024-04-10 15:32:35,281] [WARNING] [_prov_engine_client.py::170] Retrieve [GET] URL: http://cognit-lab.sovereignedge
.eu:1337/serverless-runtimes/3670
COGNIT Serverless Runtime ready after Updated!
```

Figure 3.7: Change of Serverless Runtime State from the perspective of the Device Client



FAAS_0_(service_3715) kvm
#1957 Started 1 minute ago 1 768 MB p02r11srv01 oneadmin oneadmin 2001:67c:22b8:1::2 +1 More

#1957 | FAAS_0_(service_3715)

Information	
ID	1957
Name	FAAS_0_(service_3715)
State	● BOOT_MIGRATE
Reschedule	No
Locked	-

Permissions	Use	Manage	Admin
Owner	✓	✓	☐
Group	☐	☐	☐
Other	☐	☐	☐

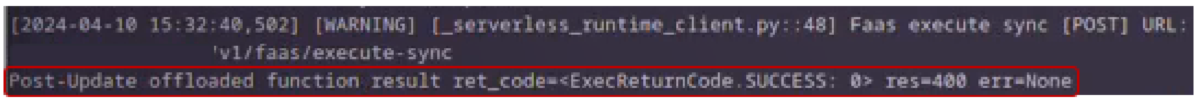
Ownership

Figure 3.8: Live migration of Serverless Runtime on host p02r11srv01

Let's define another function to offload after the update:

```python
def mult(a: int, b: int):
    print("This mult is a test")
    return a * b * a * b

result = cognit_sr.call_sync(mult, 4, 5)
```

During the live migration the client can offload the execution of the functions without any disruption of the service:

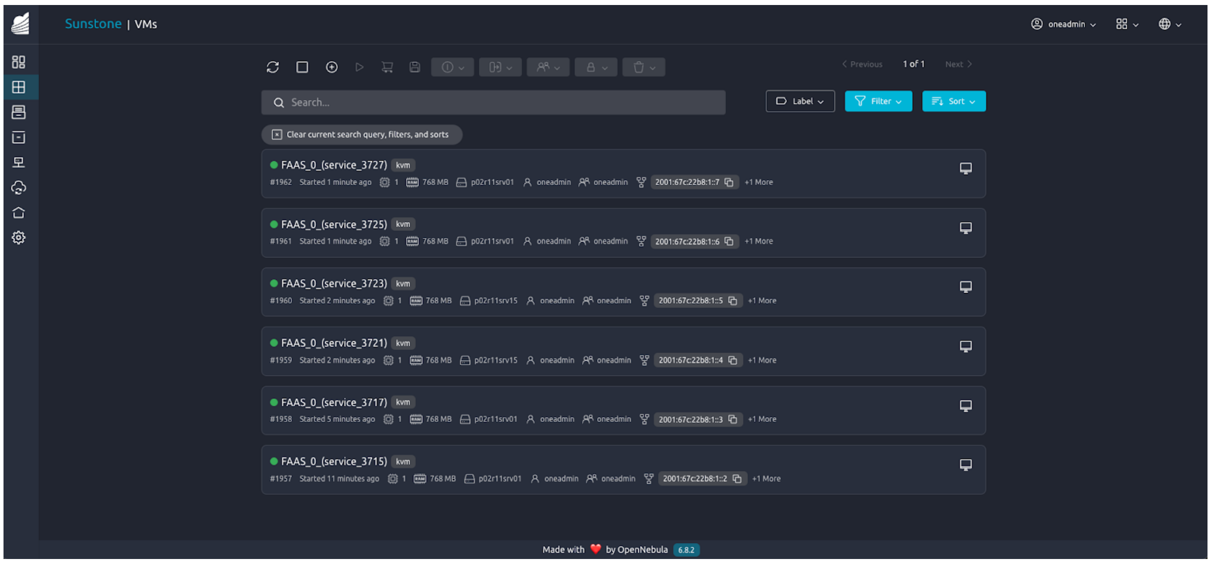


```
[2024-04-10 15:32:40,502] [WARNING] [_serverless_runtime_client.py::48] Faas execute sync [POST] URL:
                        'v1/faas/execute-sync
Post-Update offloaded function result ret_code=<ExecReturnCode.SUCCESS: 0> res=400 err=None
```

Figure 3.9: Result of the updated offloaded function (Device Client's terminal output)

3.3 Migration of Serverless Runtimes due to changes in the infrastructure

In this scenario, we assume that several Serverless Runtimes have been requested and scheduled. Some of them requested a host with ENERGY_RENEWABLE and are running on the host *p02r11srv01* as shown in the following Figure.

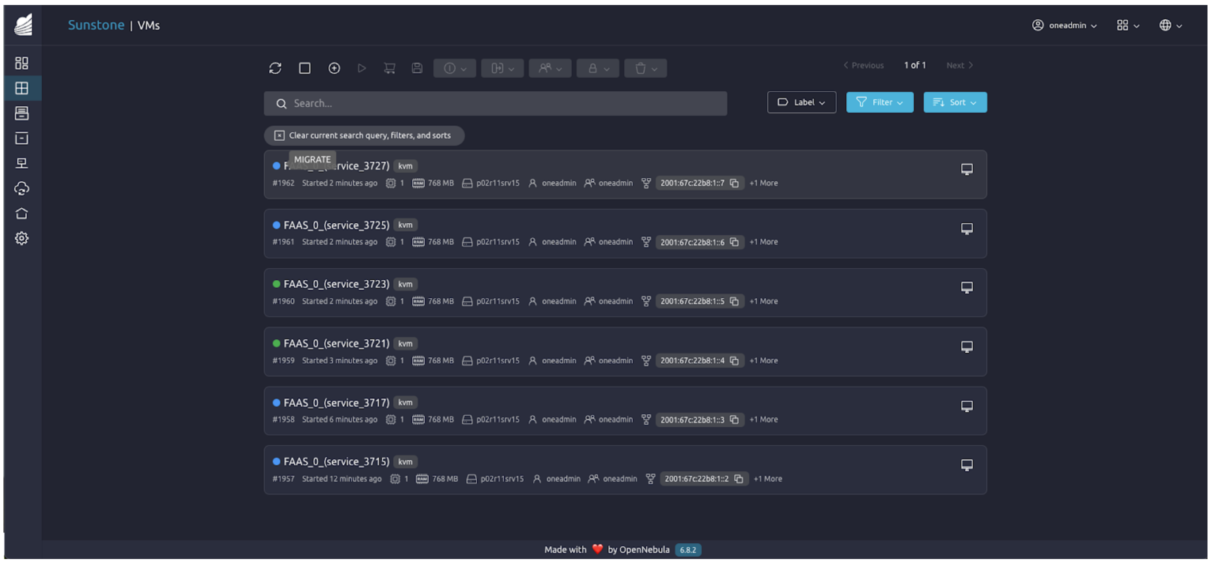


```
FAAS_0_(service_3727) kvm
#1962 Started 1 minute ago 1 768 MB p02r11srv01 oneadmin oneadmin 2001:67c:22b8:1::7 +1 More

FAAS_0_(service_3725) kvm
#1961 Started 1 minute ago 1 768 MB p02r11srv01 oneadmin oneadmin 2001:67c:22b8:1::6 +1 More

FAAS_0_(service_3723) kvm
#1960 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::5 +1 More

FAAS_0_(service_3721) kvm
#1959 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::4 +1 More

FAAS_0_(service_3717) kvm
#1958 Started 5 minutes ago 1 768 MB p02r11srv01 oneadmin oneadmin 2001:67c:22b8:1::3 +1 More

FAAS_0_(service_3715) kvm
#1957 Started 11 minutes ago 1 768 MB p02r11srv01 oneadmin oneadmin 2001:67c:22b8:1::2 +1 More
```

Figure 3.10: Active Serverless Runtimes on the COGNIT testbed, four of which are on the host *p02r11srv01*.

To simulate a change in the infrastructure, the "ENERGY_RENEWABLE=YES" label is removed from the current host (*p02r11srv01*) and assigned to the other one (*p02r11srv15*).

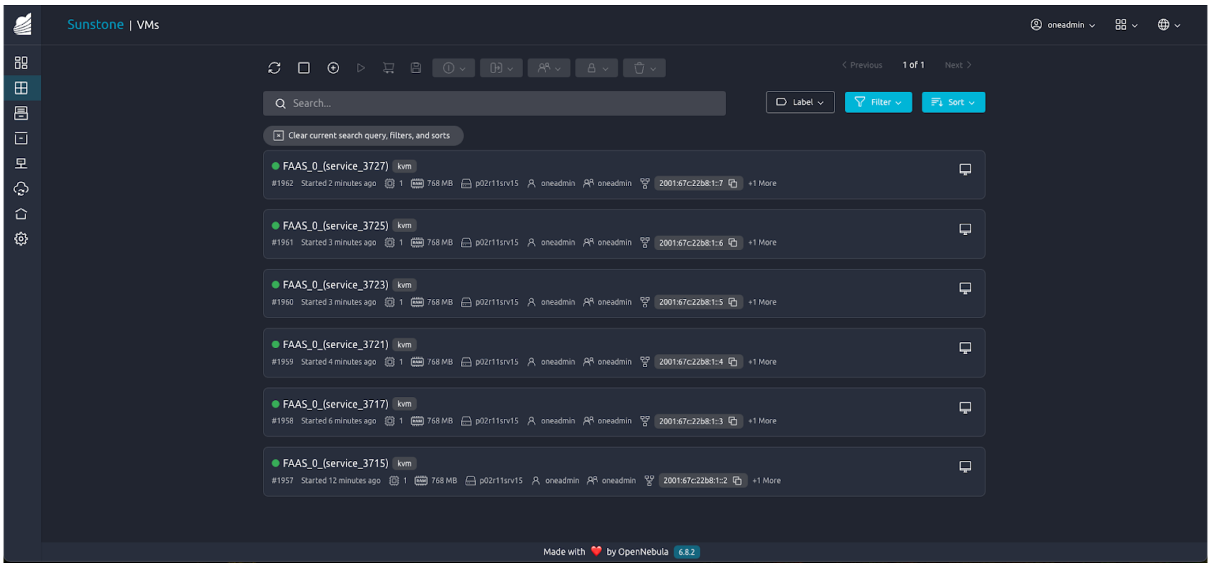
This will trigger the rescheduling of the Serverless Runtimes by the AI-Enabled Orchestrator that will migrate those ones that requested renewable energy from the host *p02r11srv01* to the host *p02r11srv15* as shown in the following figure.



```
F..._(service_3727) kvm
#1962 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::7 +1 More

FAAS_0_(service_3725) kvm
#1961 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::6 +1 More

FAAS_0_(service_3723) kvm
#1960 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::5 +1 More

FAAS_0_(service_3721) kvm
#1959 Started 3 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::4 +1 More

FAAS_0_(service_3717) kvm
#1958 Started 6 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::3 +1 More

FAAS_0_(service_3715) kvm
#1957 Started 12 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::2 +1 More
```

Figure 3.11: The result of the changes in infrastructure: the Serverless Runtimes are in the process of live migration to the host *p02r11srv15* that has now the label ENERGY_RENEWABLE=YES.

Once the live migration is finished all Serverless Runtimes are in RUNNING state as shown in the following figure:



FAAS_0_(service_3727) kvm
#1962 Started 2 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::7 +1 More

FAAS_0_(service_3725) kvm
#1961 Started 3 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::6 +1 More

FAAS_0_(service_3723) kvm
#1960 Started 3 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::5 +1 More

FAAS_0_(service_3721) kvm
#1959 Started 4 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::4 +1 More

FAAS_0_(service_3717) kvm
#1958 Started 6 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::3 +1 More

FAAS_0_(service_3715) kvm
#1957 Started 12 minutes ago 1 768 MB p02r11srv15 oneadmin oneadmin 2001:67c:22b8:1::2 +1 More

Made with ❤️ by OpenNebula 6.8.2

Figure 3.12: The Serverless Runtimes have been successively migrated and are running on host *p02r11srv15*.

4. Conclusions

This document demonstrates how to deploy the complete COGNIT Framework on a target infrastructure, in this case on public cloud resources by AWS, using the COGNIT OpsForge tool, which automatically integrates and deploys the entire COGNIT software stack.

It further demonstrates some of the capabilities of the COGNIT Framework in an operational environment, using the current COGNIT testbed hosted by RISE. In this demonstration, we specifically considered three different scenarios:

1) A device requests a Serverless Runtime and offloads a function.
2) A device updates its requirements, which triggers a migration of the Serverless Runtime.
3) Serverless Runtimes have to be migrated automatically due to changes in the underlying cloud-edge infrastructure.

Two additional incremental versions of this demo report will be released in M27 and M33.