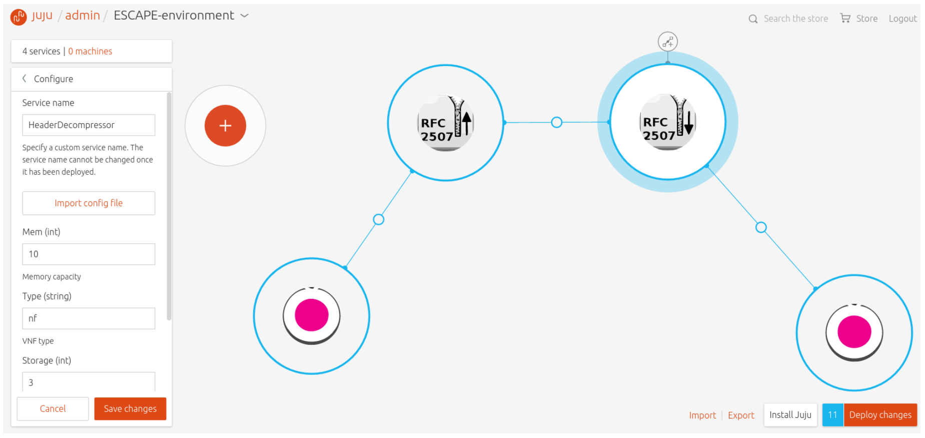
Budapest University of Technology and Economics

GUI integration into ESCAPE Service Chain Prototyping Framework

Written by: David Szabo, Attila Csoma
assistant lecturers at BME-TMIT

November 9, 2016



Service name

HeaderDecompressor

Specify a custom service name. The service name cannot be changed once it has been deployed.

Mem (int)

10

Memory capacity

Type (string)

nf

VNF type

Storage (int)

3

Figure 1: GUI main menu with some VNF

ESCAPE (Extensible Service ChAin Prototyping Environment)[1] is a general prototyping framework which supports the development of several parts of the service chaining architecture including VNF implementation, traffic steering, virtual network embedding, etc. On the other hand, ESCAPE is a proof of concept prototype implementing a novel SFC (Service Function Chaining) architecture proposed by EU FP7 UNIFY project. It is a realization of the UNIFY service programming and orchestration framework which enables the joint programming and virtualization of cloud and networking resources.

Our contribution to this project was to integrate a graphical user interface from the Juju[2] project.

0.1 GUI integration

ESCAPE uses Juju GUI[3] that is a JavaScript and HTML based web application for managing and monitoring services in a user friendly way. Juju GUI is originally created for Juju, that is a "state-of-the-art, open source, universal model for service oriented architecture and service oriented deployments". Due to the similar functionality of ESCAPE and Juju, Juju GUI is a convenient choice to be used and fine-tuned for ESCAPE as the default GUI.

With Juju GUI service definition becomes a simple three-step procedure:

i) deploy: in the top navigation menu (Fig. 1) we can explore the available VNFs (Fig. 2) and use them to create any service graph. The list of available VNFs can be loaded easily from a custom VNF Store.

ii) configure: any VNF or relationship can be fine-tuned by filling the options in its separate configuration panel displayed at the left (fig. 1).

iii) expose service: after the service graph is created with a final confirmation it is sent to the ESCAPE through the REST API.

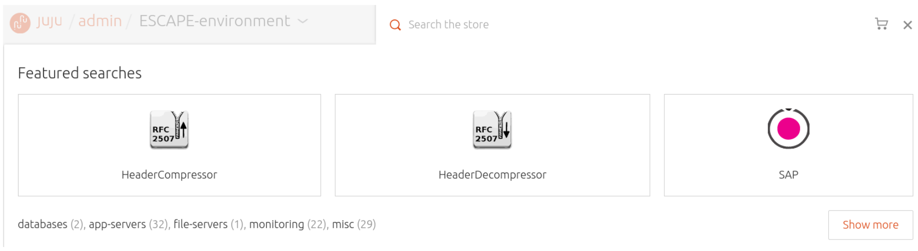

JUJU / admin / ESCAPE-environment
Search the store
Featured searches
RFC 2507
HeaderCompressor
RFC 2507
HeaderDecompressor
SAP
databases (2), app-servers (32), file-servers (1), monitoring (22), misc (29)
Show more

Figure 2: Menu of the VNF Store

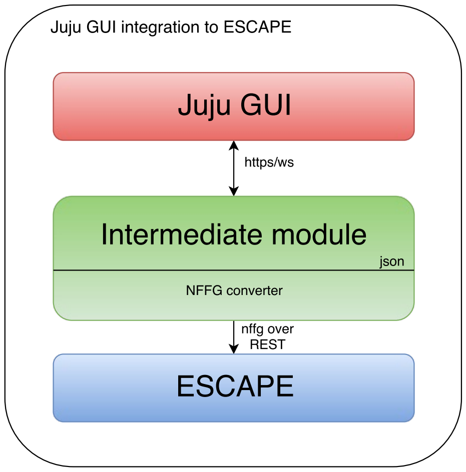


Figure 3: GUI integration to ESCAPE

After the service is started Juju GUI provides real-time information about the installed VNFs by sending queries periodically through a websocket (ws). Since Juju GUI originally is prepared to talk with Juju core an intermediate module is created between Juju GUI and ESCAPE (Fig. 3) that emulates Juju core and translates the messages appropriately.

This design choice has the advantage that only minimal modification required in the original Juju GUI codebase and enables us the continuous upgrade of the GUI as newer versions come out. To communicate with ESCAPE the intermediate module contains an NFFG converter to create proper nffg structures from the VNF chains. Communications between the intermediate module and NFFG converter taking place with json over http to achive maximal flexibility (this enables running these modules on different servers if it is required).

The intermediate module (Fig. 4) emulates a simple juju core with adaptation to ESCAPE requirements. To achive this it creates a webserver and waits for websocket and https connections on predefined ports.

GUI Connector: the main module that handles Juju GUI messages. Since Juju GUI communicates through multiple protocols (like https and websocket)

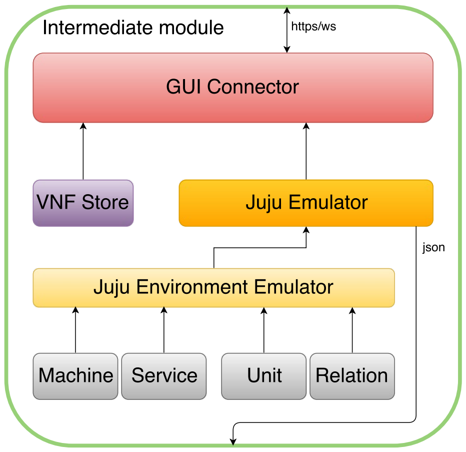


Figure 4: JujuEmulator details

it serves as a router sending messages toward the proper sub-modules.

VNF Store: the https queries for available VNFs are routed here and the VNF Store answers with the list of implemented VNFs. Currently information about VNFs are stored in a file system, but it can be easily turned into a database by defining the proper handlers.

Juju Emulator: it handles requests, sent by websocket queries, for status and update information about the VNFs. Juju GUI is able to create/delete/update links (relationships) and nodes (VNFs) in the service graph through communication with the JujuEmulator.

Juju Environment Emulator: it maintains the state of the service graph through node types (elements) listed below. Every quires and update executed in this module. Name convention inherited from the original Juju name convention to ease code maintanace, typical examples are *Machine*, *Service*, *Unit*, *Relation*.

Machine: gives information about the real devices on which the VNFs are operate.

Service: after an abstract VNF created with a specified configuration and requirement set in Juju Environment Emulator it creates a Service element.

Unit: describes the service and the machine binded together with status information (like alive, stooped, pending etc.).

Relation: describes the relationships between services (VNFs) with their metadata as well.

Bibliography

[1] https://sb.tmit.bme.hu/escape/

[2] https://jujucharms.com/

[3] https://jujucharms.com/juju-gui/