[Proceeding] NFV service dynamicity with a DevOps approach: demonstrating zero-touch deployment & operations

Original Citation:
Van Rossem, Steven; Cai, Xuejun; Cerrato, Ivano; Danielsson, Per; Nemeth, Felicián; Pechenot, Bertrand; Pelle, István; Risso, Fulvio Giovanni Ottavio; Sharma, Sachin; Sköldström, Pontus; John, Wolfgang (2017). NFV service dynamicity with a DevOps approach: demonstrating zero-touch deployment & operations. In: IFIP/IEEE International Symposium on Integrated Network Management (IM 2017), Lisbon, Portugal, May 2017. pp. 865-866

Availability:
This version is available at: http://porto.polito.it/2676998/ since: July 2017

Publisher:
IEEE

Published version:
DOI:10.23919/INM.2017.7987386

Terms of use:
This article is made available under terms and conditions applicable to Open Access Policy Article ("Public - All rights reserved") , as described at http://porto.polito.it/terms_and_conditions.html

Porto, the institutional repository of the Politecnico di Torino, is provided by the University Library and the IT-Services. The aim is to enable open access to all the world. Please share with us how this access benefits you. Your story matters.

(Article begins on next page)
NFV Service Dynamicity with a DevOps Approach: Demonstrating Zero-Touch Deployment & Operations

Steven Van Rossem∗, Xuejun Cai†, Ivano Cerrato‡, Per Danielsson§, Felicián Németh¶, Bertrand Pechenot∥, István Pelle¶, Fulvio Risso‡, Sachin Sharma∗, Pontus Sköldström∥ and Wolfgang John†
Contact email: steven.vanrossem@intec.ugent.be

Abstract—Next generation network services will be realized by NFV-based microservices to enable greater dynamics in deployment and operations. Here, we present a demonstrator that realizes this concept using the NFV platform built in the EU FP7 project UNIFY. Using the example of an Elastic Router service, we show automated deployment and configuration of service components as well as corresponding monitoring components facilitating automated scaling of the entire service. We also demonstrate automatic execution of troubleshooting and debugging actions. Operations of the service are inspired by DevOps principles, enabling quick detection of operational conditions and fast corrective actions. This demo conveys essential insights on how the life-cycle of an NFV-based network service may be realized in future NFV platforms.

I. INTRODUCTION

New network services demand higher levels of automation and optimized resource usage [1]. Our demo supports NFV service dynamicity by offering fully automated scaling and troubleshooting procedures. Flexible deployment and operations are enabled by the architecture of the UNIFY1 orchestration platform combined with an adaptive monitoring framework and a flexible yet performant Infrastructure Node.

The demo scenario showcases a situation where the Data Plane resources of a single router are not enough to handle the increased traffic load that it is facing. We demonstrate the autonomous scaling of the routing service and later introduce a bug that the framework detects and debugs on its own. During the demo, we highlight different events through deployment, assurance, and troubleshooting phases, all supported by the modular approach of our proof of concept implementation.

Implementation details and discussion on relevant lessons-learned can be found in a related IM’17 experience paper [2], while all technical background information regarding the UNIFY NFV platform are available in [3] and [4].

II. THE DEMO SETUP

The demo setup (Fig. 1) was realized on three nodes running on separate servers:

1) The Global Orchestrator Node keeps track of the available infrastructure and accepts a high-level Service Graph (SG) which describes the service as a monolithic

Fig. 1. The demo setup is based on the UNIFY NFV platform including monitoring and troubleshooting tools, implementing a service-specific scaling mechanism of an Elastic Router. Numbers 1⃝–5⃝ show the troubleshooting sequence.

router with four ports. A decomposed Network-Function-Forwarding-Graph (NF-FG) is derived from this SG, consisting of a set of service components to be deployed (right side of Fig. 1). This translation results in a Control Plane component—the Ctrl App on Fig. 1, implemented by a Ryu OpenFlow controller—and a single Data Plane component—the ovs, implemented by Open vSwitch. The NF-FG links these two components, deployable as Docker containers. It is then further mapped by the Orchestrator to the available infrastructure.

2) The Universal Node (UN) is a common hardware compute node that runs a local orchestrator software as well as a monitoring controller (MMP). It supports the deployment of different VNF types, specified in the NF-FG it receives. It is also capable of starting and configuring a set of monitoring components [5], as specified in a special NF-FG annotation (MEASURE) expressing monitoring intents for the service components. To sustain communication between the different components, a secure message bus (DoubleDecker) distributes control and monitoring information including scaling or troubleshooting triggers.

3) After deployment, the Troubleshooting Node which is also connected to the message bus, listens for notifications from the monitoring framework in the Infrastructure Node. Once an anomaly in the deployed service is detected, EPOXIDE, our specialized troubleshooting framework, automatically executes a user-defined troubleshooting process.
During the demo\(^2\), the deployment and scaling of the Elastic Router can be followed on a web-GUI exposed by the Ctrl App and the UN. The monitored data is shown in real-time on a Grafana dashboard and the Troubleshooting Node offers live output and interactive input via an Emacs-based interface.

III. THE ELASTIC ROUTER LIFE-CYCLE EVENTS

Fig. 2 depicts parts of the gathered monitoring data during the demonstration run of the Elastic Router service. To detect undesired conditions, the framework uses two metrics: the CPU load (gathered by cAdvisor), and the overload risks of the Data Plane components. The latter is a statistical estimation by RAMON, giving the risk of the byte-rate reaching an upper limit (e.g., the line-rate) \[^5\]. In the following, we walk through the main life-cycle events of the service. These events are highlighted during the demo, and are tagged in Fig. 2.

A—Deployment: The SG is passed to the Global Orche- 
s trator which maps it to the UN’s available resources and
 sends it the initial NF-FG for further deployment. The UN
 starts the two service components—the Ctrl App and one ovs
 Docker container—and sets up the necessary network links as
 described in the NF-FG. Additionally, the information in the
 MEASURE annotation of the NF-FG triggers the startup
 and configuration of the monitoring components. These, in turn,
 start to gather metrics related to the resource usage of the
deployed service—i.e. CPU load and bandwidth utilization.

B—Traffic Flow Start: After the service is fully deployed,
a traffic generator is started and traffic is sent to the four ports of
the elastic router. Initially, one Data Plane component (ovs)
is enough to forward the traffic between the four ports.

C—Scale Out Start: The traffic generator gradually in-
creases the packet rate on the four ports until a threshold for
aggregated overload risk is reached. This is detected by the
monitoring framework and a ‘scale out’ message is sent to the
Ctrl App. This event triggers the Ctrl App to generate a
new NF-FG, instructing the Orchesturator to deploy extra Data
Plane components (ovs2–5 in Fig. 1) to handle the increased
load. While scale-out is executed, monitoring is put on hold,
as shown by the gap in the curves of Fig. 2 between C and D.

During this interval, the scale-out procedure can be monitored
on the web-GUI of the Ctrl App.

D—Scale Out End: The new, scaled NF-FG also includes
a new MEASURE annotation. Thus, when scale-out is fin-
ished, the monitoring framework resumes gathering data on
the newly deployed Data Plane components, which start
forwarding traffic between the ports as depicted on the data
graph: four streams are now being generated, one for each ovs.

E—Troubleshooting: Shortly before time E we introduce
a bug to a flow table in one of the ovs instances, causing
the balanced traffic load to get disturbed. Monitoring com-
ponents (marked with ① in Fig. 1) detect this anomaly at
time E through the high variance between loads on the Data
Plans. This initiates an automatic troubleshooting process by
sending a trigger to the Troubleshooting Node (see ②).

To debug this decomposed service, the EPOXIDE troubleshooting
framework relies on special purpose debugging tools to locate
the error. A Recursive Query Engine ③ is used to check the
persistence of the error condition. It confirms the need for
debugging, hence the AutoTPG ④ flow table verification tool
is applied on each of the ovs instances. In order to properly
setup and apply each tool, the troubleshooting framework
automatically retrieves relevant information (e.g., IP and port
numbers) from different components in the UNIFY architec-
ture. Once the faulty entry in the misconfigured ovs instance is
located, it is reported to the troubleshooting operator. ⑤ marks
the only event when the operator has to interact with the
troubleshooting framework to remedy the error condition.

F—Bug Fix: The operator manually corrects the bug,
consequently the traffic loads on the different ovs components
converge again.

To sum up, this demo showcases automated deployment
of an NFV service with dynamic and autonomous service
scaling combined with programmable, zero-touch monitoring
and troubleshooting. Thus, the NFV platform successfully
adopts the principles of a DevOps approach where real time
service observability and swift debugging are key features.

ACKNOWLEDGMENT

This work is supported by UNIFY, a research project partially funded by
the European Community under the 7th Framework Program (grant agreement
no. 619609). The views expressed here are those of the authors only.

REFERENCES

art, challenges, and implementation in next generation mobile networks
(vEPC),” IEEE Network, vol. 28, no. 6, pp. 18–26, Nov. 2014.

proach: Insights from a Use-case Realization,” in IFIP/IEEE Interna-
tional Symposium on Integrated Network Management, Experience Session.

www.fp7-unify.eu/files/fp7-unify-eu-docs/Results/Deliverables


Monitoring for Service Provider DevOps,” in European Workshop on

\(^2\) A screencast of the demo scenario is at: https://youtu.be/jSxyKBzKVeS

During this interval, the scale-out procedure can be monitored
on the web-GUI of the Ctrl App.