

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

*Original*

NFV service dynamicity with a DevOps approach: demonstrating zero-touch deployment & operations / 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. - STAMPA. - (2017), pp. 865-866. ( IFIP/IEEE International Symposium on Integrated Network Management (IM 2017) Lisbon, Portugal May 2017) [10.23919/INM.2017.7987386].

*Availability:*

This version is available at: 11583/2676998 since: 2017-11-07T07:45:13Z

*Publisher:*

IEEE

*Published*

DOI:10.23919/INM.2017.7987386

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)



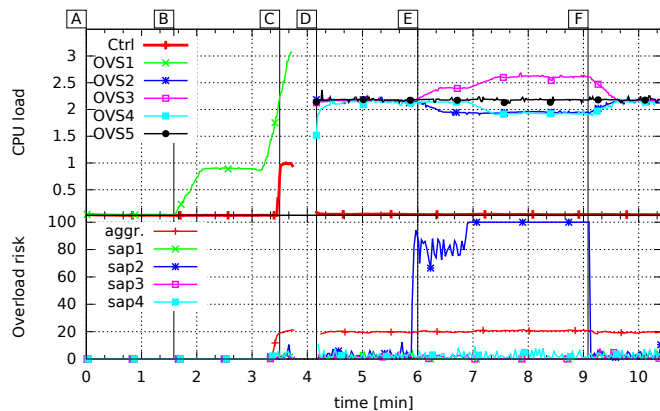


Fig. 2. The monitored service data which leads to automated scaling and troubleshooting triggers. Main events are indicated with labels A to F.

During the demo<sup>2</sup>, 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 Orchestrator 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 increases 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 Orchestrator to deploy extra Data Plane components (*ovs*2–5 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.

<sup>2</sup>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*.

**D—Scale Out End:** The new, scaled NF-FG also includes a new *MEASURE* annotation. Thus, when scale-out is finished, the monitoring framework resumes gathering data on the freshly 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 components (marked with ① in Fig. 1) detect this anomaly at time E through the high variance between loads on the Data Planes. 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 architecture. 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

- [1] H. Hawilo, A. Shami, M. Mirahmadi, and R. Asal, “NFV: state of the art, challenges, and implementation in next generation mobile networks (vEPC),” *IEEE Network*, vol. 28, no. 6, pp. 18–26, Nov. 2014.
- [2] S. Van Rossem *et al.*, “NFV Service Dynamicity with a DevOps Approach: Insights from a Use-case Realization,” in *IFIP/IEEE International Symposium on Integrated Network Management, Experience Session*. IEEE, 2017.
- [3] “Deliverable 3.5: Programmability framework prototype report,” UNIFY Project, Tech. Rep. D3.5, Jun. 2016. [Online]. Available: <https://www.fp7-unify.eu/files/fp7-unify-eu-docs/Results/Deliverables>
- [4] G. Marchetto, R. Sisto, W. John *et al.*, “Final Service Provider DevOps concept and evaluation,” *ArXiv e-prints*, vol. 1610.02387, 2016. [Online]. Available: <http://arxiv.org/abs/1610.02387>
- [5] W. John, C. Meirosu, B. Pechenot *et al.*, “Scalable Software Defined Monitoring for Service Provider DevOps,” in *European Workshop on Software Defined Networks*. IEEE, 2015, pp. 61–66.