IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-0443-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484475

# EdgeNet: the Global Kubernetes Cluster Testbed

Berat Can Şenel*,**, Maxime Mouchet*,**,
Justin Cappos†, Olivier Fourmaux*, Timur Friedman*,**, Rick McGeer‡
*Lip6-CNRS lab, Sorbonne Université, **Lincs lab, †NYU Tandon School of Engineering, ‡US Ignite

*Abstract*—**EdgeNet is a software-only distributed systems testbed in the family of PlanetLab. It is realized as a multi-tenant global Kubernetes cluster running on volunteered virtual machines. In this paper, we describe our modifications to Kubernetes which enable trusted execution with accountability on untrusted, volunteered hardware, user control of the placement of Kubernetes pods, and control of a global cluster from a single central head node.**

## I. Introduction

EdgeNet[1] is a global distributed systems testbed in the family of PlanetLab [1] and Savi [2], featuring a network of interconnected virtual machines shared between multiple competing experiments. EdgeNet is a software-only infrastructure; it is a network of VMs that run on existing local clouds. The EdgeNet control framework is a simple extension of today's de facto industry standard for cloud deployments, Kubernetes, or K8s. By using a software-only hosted virtual machine strategy, and a control framework based on common off-the-shelf (cots) technology, EdgeNet avoids the issues that plagued previous generations of distributed testbeds: hardware refresh and maintenance, and a bespoke control framework requiring its own maintenance, documentation, and user training.

A software-only opt-in framework was motivated by our previous experience with PlanetLab, PlanetLab Europe [3], and the Global Environment for Network Innovations (Geni) [4]. In all three cases, hardware acquisition and maintenance was a prohibitive cost. Moreover, keeping hardware up to date was burdensome; roughly ⅕-⅓ of the cluster's total acquisition cost should be budgeted as an ongoing refresh expense. These two factors meant that the infrastructures degraded rapidly after a few years. In contrast, software-only distributed infrastructures such as Boinc [5] and Seattle [6] were long-lived, extensible, and easy to maintain. Using a software-only infrastructure enables administrators to maintain a worldwide backbone cost-effectively, placing its VMs on servers already hosted for other purposes by participating sites.

Testbeds and testbed federations such as PlanetLab, Emulab [7], G-Lab [8], Geni, OneLab [3], Fed4Fire [9], Fit IoTLab [10], Grid'5000 [11], CloudLab [12] use, or used, bespoke control frameworks, which entails a number of challenges. First, the framework has to be developed and maintained by a small group of people, ported to new hardware and environments; any tools and technologies have to be developed for or ported to the framework by the same small group; training materials for users have to be developed and maintained (and the Geni tutorials, for example, largely consisted of training in how to use the Geni framework). By using the world's most popular control framework, EdgeNet has enlisted the worldwide community of K8s users and developers in solving these problems for us. The underlying framework is constantly improving thanks to the efforts of tens of thousands of users and developers worldwide.

[1]EdgeNet main website is at https://edge-net.org, and its source code is available at https://github.com/EdgeNet-project.

However, adopting K8s presented a number of challenges. K8s was designed to control homogeneous worker nodes on a single lan, where few of the pods accessed the public Internet, where the pods and cluster were mutually trusting, where there was a single tenant, and where the intra-cluster latency was on the order of microseconds.

In contrast, EdgeNet is an infrastructure where the worker nodes are located in different environments (data centers, universities, ...), where every tenant is conducting experiments on the public Internet, where the cluster cannot trust the experiment software and the software cannot trust the cluster, where each worker node is shared by multiple tenants, and where inter-pod latencies and those between control plane and worker nodes are on the order of tens of milliseconds. In this paper, we describe how we used extant K8s facilities and, where appropriate, extended K8s to address some of these challenges.

## II. Multitenancy

A multitenant testbed supports simultaneous experiments by different users. Containerized operating systems allows those experiments to run side-by-side on the same host. PlanetLab was built on lightweight container technologies: initially Linux-VServer, then migrating to LXC. One user's container on a node was a *sliver*, and a user's set of slivers across all nodes was a *slice*. The slice-based model is now common to multitenant testbeds.

EdgeNet uses today's industry standard for lightweight containerization: Docker. The equivalent of a sliver is a K8s *pod*, a grouping of one or more containers. This has three advantages over earlier PlanetLab technology. First, Docker has built-in support for a wide range of Linux OSes, whereas each PlanetLab node could only host one OS, and one (soon outdated) version of that OS. Second, use of pods allows an experiment to be built from multiple containers, allowing each container environment to be built to match the software component that it supports. Third, Docker containers are easily built in one location, then deployed to nodes via K8s. By contrast, software typically had to be built directly on each

PlanetLab node, and there was no PlanetLab-native mechanism for orchestrating the process.

In addition to the multitenancy features that EdgeNet automatically inherits from its use of Docker, we have extended the K8s user model to support users from multiple different institutions. As other testbeds have done, EdgeNet follows PlanetLab's model of having the central administrators approve recognized researchers, and delegating the authorization of other users to those researchers. We have leveraged K8s namespaces to do so, as these provide isolation. EdgeNet gives each researcher control over their own namespace, and, because namespaces are hierarchical, this allows a researcher to in turn delegate control over a subsidiary namespace to another researcher, who then authorizes the members of their team. Similarly, a professor can delegate to a teaching assistant, who in turn enables student accounts.

## III. Worker Pod Placement

EdgeNet's value as compared to vanilla K8s is its ability to deploy containerized software to a widely distributed set of nodes rather than to nodes that are all grouped together in a centralized datacenter.

In K8s, pod creation is orchestrated by *deployment* objects. Deployments create *replica sets* which themselves ensure that the desired number of containers is running at any given time. If a node fails, new containers are automatically scheduled on new nodes. Furthermore, a deployment can be updated to change the containers version, or to change the number of containers.

EdgeNet extends K8s deployments with *selective deployments*. Selective deployments currently allow the user to choose the geographical placement of the pods, at the continent, country, state/region, and city resolutions. The system searches the latitude & longitude information of nodes in a GeoIP database to pick up the relevant nodes. If a node goes down in a requested location, the affected containers will be scheduled on another node in the same location.

Let us note that for some experiments, it may be more practical to to choose nodes based on their network location (e.g. their autonomous system) rather than their geographical location. Future versions of EdgeNet will take into account this need.

## IV. Ease of Use

The goal in user experience design in EdgeNet is to offer as transparent an overlay on K8s as possible. Our response to the question "How do I use EdgeNet?" is *"It's just Kubernetes"*, which instantly tells the questioner how to use it, and provides access to a vast array of online tutorials and training materials.

Every action in EdgeNet can be performed with the standard K8s `kubectl` tool: registering a new node, creating an user, running an experiment, etc. We have used the K8s *custom resource* mechanism to achieve this. When a custom resource is defined for K8s, its API is automatically extended to accommodate it. `kubectl` commands remain the same, the only difference being that additional arguments are possible.

Because EdgeNet will have some users who are less comfortable with `kubectl`, we have also provided a web portal. This is useful to researchers who will not themselves be running experiments, but who will be authorizing members of their team to do so. For these individuals, EdgeNet offers the possibility to log in with a classic username and password and managing users and nodes through via the web.

## V. Platform Status

EdgeNet is currently made of 40 nodes worldwide including 5 in Europe, 1 in Brasil, 1 in Australia, and the others in the United States. 54 users are registered and it has supported multiple experiments over the past year: The *CacheCash* project at NYU Tandon School is a blockchain-based CDN where end users participates in content distribution; it has been deployed on 30 EdgeNet nodes to perform performance measurements. *Diamond-Miner*, a network topology discovery tool built at Sorbonne Université, has been deployed on 7 EdgeNet nodes to conduct internet-wide measurements. The *Cyberlab Honeypot* experiment at the University of Ljubljana has used EdgeNet to expose fake SSH servers on the internet and record malicious activities. The *Darknet Watch* experiment at the University College Dublin is using EdgeNet to perform measurements on the I2P anonymous network. Finally, EdgeNet is running the *M-Lab NDT* (Network Diagnostic Tool) client to measure download and upload speeds.

## Acknowledgment

EdgeNet got its start thanks to an NSF Eager grant, and now benefits from a VMware Academic Program grant and a French Ministry of Armed Forces cybersecurity grant.

## References

[1] L. Peterson, A. Bavier, M. E. Fiuczynski, and S. Muir, "Experiences building PlanetLab," in *Proc. OSDI*, 2006.
[2] A. Leon-Garcia and H. Bannazadeh, "SAVI testbed for applications on software-defined infrastructure," in *The GENI Book*. New York: Springer, 2016, pp. 545–562.
[3] S. Fdida, T. Friedman, and T. Parmentelat, "OneLab: An open federated facility for experimentally driven future internet research," in *New Network Architectures*. Springer, 2010, vol. 297, pp. 141–152.
[4] R. McGeer, M. Berman, C. Elliott, and R. Ricci, Eds., *The GENI Book*. Springer, 2016.
[5] D. Anderson, "BOINC: A system for public-resource computing and storage," in *Proc. GRID*, 2004.
[6] J. Cappos, I. Beschastnikh, A. Krishnamurthy, and T. Anderson, "Seattle," *ACM SIGCSE Bull.*, vol. 41, no. 1, pp. 111–115, Mar. 2009.
[7] R. Ricci and t. E. Team, "Precursors: Emulab," in *The GENI Book*. New York: Springer, 2016, ch. 2, pp. 19–33.
[8] P. Müeller and S. Fischer, "Europe's mission in next-generation networking with special emphasis on the German-lab project," in *The GENI Book*. New York: Springer, 2016, ch. 21, pp. 513–544.
[9] P. Demeester, P. Van Daele, T. Wauters, and H. Hrasnica, *Fed4FIRE: The largest federation of testbeds in Europe*. River Publishers, 2016.
[10] E. Fleury, N. Mitton, T. Noel, and C. Adjih, "FIT IoT-LAB: The Largest IoT Open Experimental Testbed," *ERCIM News*, no. 101, p. 4, Apr. 2015.
[11] R. Bolze, F. Cappello, E. Caron, M. Daydé, F. Desprez, E. Jeannot, Y. Jégou, S. Lanteri, J. Leduc, N. Melab, G. Mornet, R. Namyst, P. Primet, B. Quetier, O. Richard, E.-G. Talbi, and I. Touche, "Grid'5000: A large scale and highly reconfigurable experimental grid testbed," *IJHPCA*, vol. 20, no. 4, pp. 481–494, Nov. 2006.
[12] R. Ricci, E. Eide, and The CloudLab Team, "Introducing CloudLab: Scientific infrastructure for advancing cloud architectures and applications," *USENIX ;login:*, vol. 39, no. 6, Dec. 2014.