TEAS
Internet-Draft
Intended status: Informational
Expires: 4 September 2025

L. M. Contreras
Telefonica
X. Liu
Alef Edge

3 March 2025

DC aware TE topology model
draft-llc-teas-dc-aware-topo-model-04

Abstract

This document proposes the extension of the TE topology model for including information related to data center resource capabilities.

Status of This Memo

This Internet-Draft will expire on 4 September 2025.

Copyright Notice

Table of Contents

1. Introduction

More and more service providers are deploying cloud computing facilities in order to host different kinds of services and applications. Such facilities can be generally referred as Datacenter Points of Presence (DC-PoPs). Those DCs will consist of a number of servers and networking elements for connecting all of them with the transport network. Depending on the number of servers in the data center, there will be distinct capabilities in terms of CPUs, memory and storage available for deploying and running the aforementioned services.

In such distributed and interconnected DC-PoPs, both computing and topological information are of interest for determining the optimal DC where to deploy a given service or application.

This document proposes a DC-aware extension for the topology model.

2. Datacenter information

The relevant information for datacenter capabilities can be described in different ways. One potential manner is to describe resource capabilities such as CPU, memory, storage, etc. This can be done in terms of total, used and free capacity for each of the parameters of interest. Cloud management systems allow to obtain such kind of information. For instance, in the case of Kubernetes it is possible to retrieve information about the total and allocatable resources in a compute node. Alternatively, in the case of OpenStack it is possible to collect information about the total resources and those currently in use from such total.

Another form of populating the information is by describing those resource capabilities as a bundled, usually referred as quota or flavor. Well known cloud computing providers like Amazon Web Services, Microsoft Azure or Google Cloud Platform follow such schema, bundling CPU, RAM and storage units as flavors. In the case of Amazon Web Services the bundle is known as “instance type”, while in Microsoft Azure is termed “virtual machine size” and in Google Cloud Platform it is called “machine type”.

Additional information to consider in both cases could refer to the management capabilities of the compute infrastructure, such as hypervisor details or virtualization technologies available.

The recent trend of cloud-native approach for the instantiation and deployment of service functions has positioned Kubernetes as the de-facto standard to manage containerized software in data centers, but it is also increasingly being used by telecommunication operators to manage compute resources at the edge.

Finally, all can be complemented with information related to the networking details for reaching the aforementioned compute capabilities (IP addresses, bandwidth, etc).

3. Model structure

According to the distinct approaches for managing cloud-based resources different options could exist.

3.1. Hypervisor-based cloud solutions

A model structure for hypervisor-based cloud solutions (e.g., OpenStack) can be described in the following manner.

```
      module: ietf-openstack-info
          +--rw dcpop
             +--rw dcpop-id?   string
             +--rw dc* [id]
                +--rw openstack
                   +--rw compute-nodes
                   |  +--rw node* [name]
                   |     +--rw name                string
                   |     +--rw vcpus
                   |     |  +--rw total            uint64
                   |     |  +--rw allocated        uint64
                   |     |  +--rw used             uint64
                   |     +--rw memory
                   |     |  +--rw total            uint64
                   |     |  +--rw allocated        uint64
                   |     |  +--rw used             uint64
                   |     +--rw instances
                   |        +--rw max-instances    uint32
                   |        +--rw running          uint32
                   +--rw instances
                      +--rw instance* [id]
                         +--rw id                 string
                         +--rw name               string
                         +--rw project-id         string
                         +--rw vcpus
                         |  +--rw allocated       uint64
                         |  +--rw limit           uint64
                         |  +--rw used            uint64
                         +--rw memory
                         |  +--rw allocated       uint64
                         |  +--rw limit           uint64
                         |  +--rw used            uint64
                         +--rw status
                            +--rw state           enumeration
                            +--rw conditions*     string
```

3.2. Container-based cloud solutions

A model structure for container-based cloud solutions (e.g., Kubernetes) can be described in the following manner.

```
module: ietf-kubernetes-info
       +--rw dcpop
          +--rw dcpop-id?   string
          +--rw dc* [id]
             +--rw kubernetes
                +--rw nodes
                |  +--rw node* [id]
                |     +--rw name               string
                |     +--rw cpu
                |     |  +--rw capacity        uint64
                |     |  +--rw allocatable     uint64
                |     |  +--rw usage           uint64
                |     +--rw memory
                |     |  +--rw capacity        uint64
                |     |  +--rw allocatable     uint64
                |     |  +--rw usage           uint64
                |     +--rw pods
                |        +--rw max-pods        uint32
                |        +--rw running-pods    uint32
                +--rw pods
                   +--rw pod* [id]
                      +--rw namespace         string
                      +--rw name              string
                      +--rw cpu
                      |  +--rw request        uint64
                      |  +--rw limit          uint64
                      |  +--rw usage          uint64
                      +--rw memory
                      |  +--rw request        uint64
                      |  +--rw limit          uint64
                      |  +--rw usage          uint64
                      +--rw status
                         +--rw phase          enumeration
                         +--rw conditions*    string
```

4. Security and operational considerations

The data-model in this document does not have any security implications. The model is designed to be accessed via NETCONF [RFC6241], thus the security considerations for the NETCONF protocol are applicable here.

5. Informative References

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

Acknowledgements

The work of L.M. Contreras has been partially funded by the European Union under the Horizon Europe projects NEMO (NExt generation Meta Operating system) grant number 101070118, and CODECO (COgnitive, Decentralised Edge-Cloud Orchestration), grant number 101092696.

Young Lee contributed to initial versions of this draft.

Authors' Addresses

Luis M. Contreras
Telefonica
Ronda de la Comunicacion, s/n
28050 Madrid
Spain
Email: luismiguel.contrerasmurillo@telefonica.com
URI: http://lmcontreras.com

Xufeng Liu
Alef Edge
Email: xufeng.liu.ietf@gmail.com