Ontology for OpenStack Service Architectures

Ales Komarek
Faculty of Informatics and Management
University of Hradec Kralove
Czech Republic
Email: ales.komarek@uhk.cz

Jakub Pavlik
Faculty of Informatics and Management
University of Hradec Kralove
Czech Republic
Email: jakub.pavlik.7@uhk.cz

Vladimir Sobeslav
Faculty of Informatics and Management
University of Hradec Kralove
Czech Republic
Email: vladimir.sobeslav@uhk.cz

***Abstract*—**

This paper explains how ontology can be used to model various OpenStack architectures. OpenStack is the largest open source cloud computing IaaS platform. It has been gaining wide spread popularity among users as well as software and hardware vendors over past few years. It's a very flexible system that can support a wide range of virtualization scenarios at any scale.

In our work we propose a formalization of OpenStack architectural model that can be automatically validated and serve suitable meta-data to configuration management tools. The OWL-DL based ontology defines service components and their relations and provides foundation for further reasoning. Provided models can support simple all-in-one architecture as as well as large architectures with service components in HA setup.

I. Introduction

OpenStack is the largest open-source cloud computing platform today. Many companies participate to its code, extend core functions and write new service backends to fit their business goals. The actual system consists of many components designed with plugin architecture that allows custom implementations for various service backends. These components can be combined and configured to match available software and hardware resources and real use-case needs.

Each implementation has its own component combination and use some form of configuration management tool to enforce the service states on designated servers and possibly other network components. These tools require data that covers configuration of all components. Detecting component inconsistencies by hand is painful and time consuming process.

We propose a formalization of OpenStack service architecture model, based on the approaches developed in classic knowledge representation domain, especially Service-Oriented Architecture by OpenGroup. Component definition is encoded in an ontology using the standard OWL-DL language, which enables sharing of knowledge about configurations across various systems. Reasoning can be used on the specification to automate validation of configuration changes.

When dealing with hundreds of components with thousands of properties and relations, keeping track of changes throughout its life cycle is very challenging. Current approaches are ad hoc, even OpenStack Fuel has severe limitations, there exists no standard for specifying common OpenStack architectural model. The question how to convert the proposed OWL-DL schema to metadata format that configuration management tools can process is discussed. We are working on external node classification service that uses graph database to serialize the OWL ontology with REST API that configuration management tools can use as metadata provider. This can streamline the process of adopting new services and service backends in predictable manner.

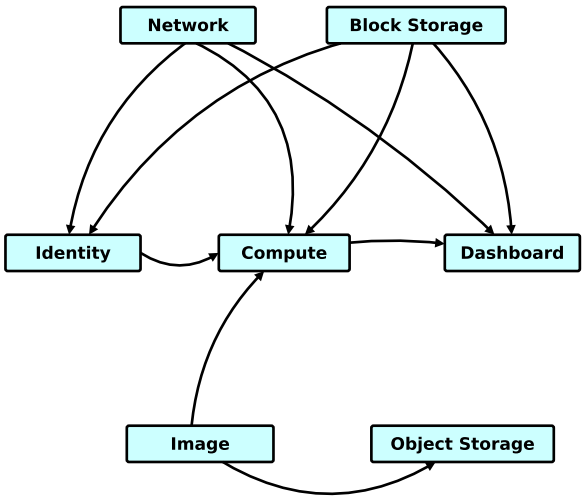

Fig. 1. Logical Model of Havana OpenStack

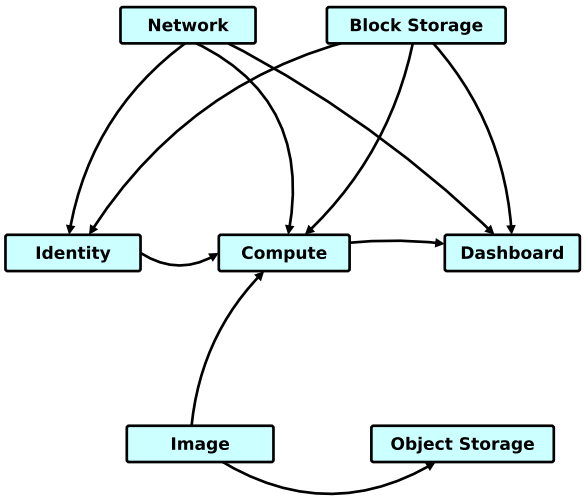

Fig. 2. Logical Model of Icehouse OpenStack

A. IaaS Solutions

B. Use Cases

C. Infrastructure Modeling

II. Service Architecture Models

Jak funguje IaaS ve smyslu deploye masiny z pohledu controlleru - zavola scheduler, ten comupte, ten pak glance, pak pripadne cinder a neutron a pusti boot, kdyz to ma ready. a tim padem provoz.

obzrazel

A. Architectural Level

OpenStack is complete Infrastructure as a Service platforms. It allows to create virtual servers on virtual networks using virtual block devices. These core services are followed by growing number of services covering for example telemetry, orchestration or data processing. All services within OpenStack architecture have pluggable backends. This allows vendors to develop plugin for their resources, that can be accessed and managed by the OpenStack API.

Figure X shows the basic configuration of OpenStack in Icehouse version.

Klasicky logicky model openstack architektury

2) OpenStack architecture moduls

Rozebrat services a pedstavit modularitu a vendor plugins, drivers

Database

Message queue

Time service

Identity - Keystone

Image - Glance

Compute - Nova

Network - Neutron

Volume - Cinder

B. IaaS Controller Support Services

This section covers the

Real implementations of Architectural models

1) High Availability Services: Cluster software - corosync/-pacemaker - keepalived

2) Communication Services: RPC - rabbitmq - qpid - 0mq

3) Database Services: Database - mysql/galera - postgressql/xtradb

4) Time Synchronization Services: time - ntp

C. Iaas Controller Core Services

APIs

Pluggable backends

1) Identity Service: Keystone is an OpenStack project that provides Identity, Token, Catalog and Policy services for use specifically by projects in the OpenStack family.

Keystone - file - sql - ldap

2) Image Service: OpenStack Image service (code-named Glance) provides services for virtual disk images. Compute service uses image service to get the starting image of the virtual server.

OpenStack Image service handles variety of disk image formats, including Raw, Machine (kernel/ramdisk outside of image, also known as AMI), VHD (Hyper-V), VDI (VirtualBox), and qcow2 (Qemu/KVM).

- dir - swift - s3

3) Compute Service: OpenStack Compute service (codename Nova) is designed to provision and manage large networks of virtual machines, creating a redundant and scalable cloud-computing platform. It provides the software required to orchestrate compute instances through using libvirt or or other virtualization client libraries. OpenStack Compute service is both hardware and hypervisor agnostic, currently supporting a variety of standard hardware configurations and major hypervisors.

- kvm - qemu - docker - hyper-v

4) Network Service: Neutron

- flat - ovs-gre/vxlan - sdns

5) Volume Service: Cinder

- lvms - sans

Rzn zpsoby nasazen ukzky reln architektury - promapovat ve 4 na ontologii

D. Hardware matters

Why we choose different openstack setups

Lab1 - 20 hypervisors, kvm, ovs-gre, local hdd Lab2 - 4 hypervisoers, kvm, sdn-contrail, san Lab3 - 5 hypervisors, ...

III. OpenStack Deployment Tools

There are many ways how to deploy OpenStack infrastructure which are more or less automated. Some of them require to fill in answer files, some configuration files. Some tools have graphiceal user interface and allow to provision entire hardware infrastructure as some just configure the services on the provisioned servers.

Model je popsanej dokumentem a nen to iteln, automatizace. Nen validita modelu. Chyby se debuguj na rovni reality.

A. Development

For testing and developing OpenStack ...

1) PackStack:

2) Devstack:

B. Production

1) Fuel:

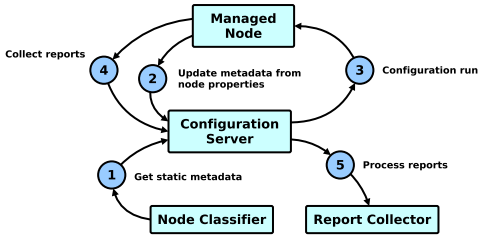

Fig. 3. Configuration Management deployment cycle

2) Foreman:

IV. OpenStack Platform Ontology

Let's start what the ontology is, the short is answer:

An ontology is a specification of a conceptualization.

The broader answer may be:

In the context of computer and information sciences, an ontology defines a set of representational primitives with which to model a domain of knowledge or discourse. The representational primitives are typically classes (or sets), attributes (or properties), and relationships (or relations among class members). The definitions of the representational primitives include information about their meaning and constraints on their logically consistent application.

There formal definition exists

Servers (physical and virtual)

Core Infrastructure Services (DNS,DHCP,NTP, image management)

Storage (NAS and SAN)

Network (Routers, Switches, Firewalls, Load Balancers)

Facilities (Power, Cooling, Space)

A. Standard Ontologies

The ontologies define the relations between terms, but does not prescribe exactly how they should be applied.

1) Service-Oriented Architecture: The SOA ontology specification was developed in order to aid understanding, and potentially be a basis for model-driven implementation of software systems

The ontology is represented in the Web Ontology Language (OWL) defined by the World-Wide Web Consortium (W3C). OWL has three increasingly expressive sub-languages: OWL-Lite, OWL-DL, and OWL-Full [OWL]. This ontology uses OWL-DL, the sub-language that provides the greatest expressiveness possible while retaining computational completeness and decidability.

The ontology contains classes and properties corresponding to the core concepts of SOA. The formal OWL definitions are supplemented by natural language descriptions of the concepts, with graphic illustrations of the relations between them, and with examples of their use. For purposes of exposition, the ontology also include.

2) OSLC Configuration Management:

B. Serialization Formats

Zpsoby serialiazce ontologie

1) XML Documents: RDF format

2) Graph databases: Preserving RDF format, just very different implementation

je to servica, tzn overhead oproti xml filu, ale zas ma api atd ...

3) Hierarchical: Subject (id) or property driven

C. Comparison

Srovnn jednolivch formt pro ontologii pro openstack een - bezpenost, rychlost, integrace

- speed - parsing / scaling

- integration, maintenance costs

- security issues

V. Ontology Usage

A. Integrity Validation

Deployment bez implementanch chyb.

1) Model Validation: Validace celho een vi high level modelu.

B. External Node Classification

Mapovn vybranho formtu na OpenStack

(HA architektura SDN controller)

1) Puppet ENC:

2) SaltStack ENC: Reln pnos celho een

VI. Conclusion

Ontologick reprezentace prosted, kter je vhodn pro agentov prosted, aby bylo mon provdt autonomn rozhodnut.

In our work we created ontology of OpenStack services.

Vytvoen idelnho prosted.

Zpsoby penesen ontologie do realizace

A. Future work

Low level realizace prostednictvm configuration management tools a jejich transformac z modelu.

Acknowledgment

This paper is published thanks to the financial support of the European Operational Programme Education for Competitiveness project INDOP CZ.1.07/2.3.00/45.0014 and UHK specific research project no. 2101.

REFERENCES

[1] *NIST. The NIST Definition of Cloud Computing* http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf

[2] Ivan Ivanov, Marten van Sinderen and Boris Shishkov, editors. *Cloud Computing and Services Science* Springer Science, 978-1461423256, New York, USA, 2012

[3] *OpenStack. Fuel Wiki* https://wiki.openstack.org/wiki/Fuel

[4] *CloudStack. Open Source Cloud Computing: Apache CloudStack* http://cloudstack.apache.org/about.html

[5] *Fedora Project. OpenStack devstack* http://fedoraproject.org/wiki/OpenStack_devstack

[6] *reclass. Recursive external node classification* http://reclass.pantsfullofunix.net/

[7] *CFEngine. FCEngine 3.5 Dcoumentation* https://cfengine.com/docs/3.5/index.html

[8] *Puppet Labs. Creating Hierarchies* http://docs.puppetlabs.com/hiera/1/hierarchy.html

[9] *The Foreman. The Manual: Compute Resources* http://theforeman.org/manuals/1.4/index.html# 5.2ComputeResources

[10] *Configuration Management Resource Definitions* http://open-services.net/wiki/configuration-management/Configuration-Management-Resource-Definitions/

[11] *Configuration Management Resource Definitions* http://www.w3.org/2004/OWL

[12] *Web Ontology Language (OWL)* http://www.w3.org/2004/OWL