Ontology for OpenStack 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 covers configuration for various OpenStack architectures. OpenStack is the largest opensource cloud computing platform. It has gaining wide spread popularity among user as well as vendors. It is a very flexible system that can provision a wide range of virtualization scenarios.

In our work we propose a formalization of OpenStack architectural model that can be validated and served to configuration management tools as a node classifier. The OWL-DL based ontology components and their relations serve as foundation for further reasoning and medatada provision.

I. INTRODUCTION

OpenStack is the largest opensource cloud computing platform. Many companies participate to its codebase and extend by writing new plugins and improving the core. The actual system consists of many components designed with plugin architecture to allow custom implementations for various backends. These components can be combined and configured to match available hardware resources and use case needs.

Each implementation has its own component setup and use configuration management tools like Foreman, Salt or Fuel to enforce the service states on designated resources. These tools require complete metadata set covering configuration of all compontents. Detecting component inconsistencies by hand is painful and time consuming process.

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

When dealing with hundreds of components, keeping track of changes throughout its life cycle is very challenging. Current approaches are ad hoc and proprietary, and 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.

1) Introduction - pro OpenStack - protoe komunita 500000 vvoj, tisce firem, nrst kdu za posledn dobu stacalytics.com vendor lockin, scalability from notebook to thousans of servers like CERN - pedstavit OpenStack jako systm s rolemi, konfiguracemi, komponenty, drivery. jina instalace per use case. Nexexistuje univerzln instalace. Vysoka komplexita, services

Jak vytvoit high level model (logick schma) architektury? a penst ji do low level design realizace? Jak sprvn definovat architekturu na zklad hw infrastruktury a target use case? Jak cel proces deploymentu automatizovat?

A. *IaaS Solutions*

B. *Use Cases*

C. *Infrastructure Case Modeling*

II. ARCHITECTURAL 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. Solution level

Real implementations of Architectural models

1) IaaS controller support: Cluster software - corosync/-pacemaker - keepalived

Database - mysql/galera - postgressql/xtradb

RPC - rabbitmq - qpid - 0mq

time

2) Iaas Controller services: Pluggable backends

Keystone - file - sql - ldap

Glance - dir - swift

Nova - kvm - qemu - docker - hyper-v

Neutron - flat - ovs-gre/vxlan - sdns

Cinder - lvms - sans

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

C. 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:

2) Foreman:

IV. Ontology of OpenStack System

A. Metadata Standards

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. External Node Classification

Mapovn vybranho formtu na OpenStack (HA architektura SDN controller)

1) Model Validation: Validace celho een vi high level modelu. Deployment bez implementanch chyb.

Reln pnos celho een

VI. Conclusion

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

Vytvoen idelnho prosted.

Zpsoby penesen ontologie do realizace

Future works - 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/