ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

A Survey on Cloud Application Portability

K. Naren Chandra [1], V. Neelanarayanan[2]

P.G. Student, School of Computer Science Engineering, VIT University, Chennai, India[1]

Associate Professor, School of Computer Science Engineering, VIT University, Chennai, India[2]

ABSTRACT: Cloud computing has quickly developed and spread in a previous couple of years, with constantly new administrations and functionalities being offered by suppliers keeping in mind the end goal to increase bigger market divisions. This has brought on, as a rule, a great deal of trouble and perplexity to clients who have been regularly subjected to the vendor lockin phenomenon, because of interoperability and portability issues frequently emerging among various Cloud suppliers. In this paper, we give a brief prolong to the essential definitions of Cloud Computing, portability, and interoperability and we additionally portray an arrangement of built up utilize cases. Every one of these ideas is mapped to a multi-dimensional space, which is utilized to order both definitions and utilize cases.

KEYWORDS: cloudify director, TOCSA, portability, Agent, REST API etc.,

I. Introduction

Cloud portability is the ability to move applications and information starting with one cloud computing environment then onto the next with irrelevant unsettling influence.Cloud portability enables the migration of cloud services from one cloud provider to another or between a public cloud and a private cloud. Clientsor organizations would prefer not to be bolted into a solitary supplier alternative and might need to alleviate hazard and increment their adaptability through the capacity to move their applications and data between cloud service providers as their computing workloads fluctuate and the business necessities change as the need emerge they might want the flexibility to move among thedifferent cloud deployment models by possibly moving to public clouds as well as between public, private and hybrid clouds.

The main reason for cloud portability are:

- Economic Reasons: The clients of the cloud would pick up assurance over their interests in their application developments.
- Technical Reasons: All through a lifetime of an application, a point may come at which outer assets are required from a public cloud, thus it should be redeployed from a private cloud.

II. CLOUD PORTABILITY IN THE ENTERPRISE

There have been distinctive endeavours and ways to deal with overseeing manage this test on different levels of the following Fig. 1.

A. NESTED VIRTUALIZATION

The primary preferred standpoint of this approach is that it can be connected to any application today that is as of now bundled into a VM and following fig. 2

- Pros:
  - Empowers consistent move of existing applications between clouds
- Cons:
  - This model naturally needs auto-scaling, self-healing, and modern monitoring capabilities
  - There is execution overhead and debasement issues.
  - There is a high reliance on outsider hypervisor suppliers as the basic hypervisor between clouds.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

B. CONTAINERS

Unlike virtual machines that need to experience a hypervisor layer, containers run as service within the operating system, and accordingly, this gives a fundamentally lighter and more convenient bundling model and Fig. 3.

- Pros:
  - Have proven to be simple to implement and are quickly gaining momentum and popularity.
- Cons:
  - This model additionally innately needs auto-scaling, self-healing, and modern monitoring capabilities.
  - Transforming existing application into containers is often not a trivial undertaking.

C. API COMPATIBILITY

Programming interface similarity fundamentally shows the capacity to give a typical API that will work over all clouds. There are fundamentally two groups of cloud compatibility APIs. The first are API compatibility frameworks such as JCLouds or LibCloud and Fig. 4.

- Pros:
  - Compatible with all the cloud ecosystem frameworks of the selected cloud.
- Cons:
  - There is a constrained similarity as far as the quantity of clouds supported in this option and those are as of now very few
  - Mapping API semantics can often be confusing and complex.

Applications
Portability?
vmware vSphere
openstack
BARE METAL Cloud

Fig. 1: cloud portability

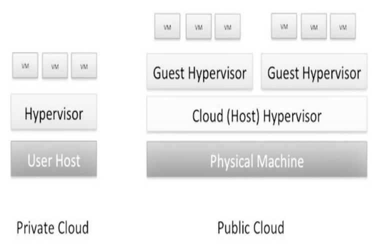


Fig. 2: Nested Virtualization

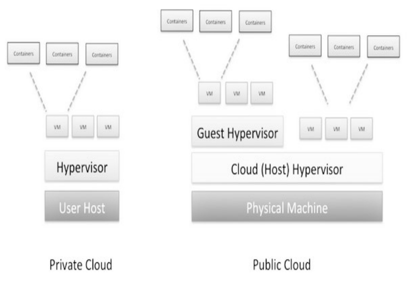


Fig. 3: Containers

D. PAAS

PaaS gives a layer of reflection the empowers clients to send applications specifically to the cloud without being presented to the fundamental cloud infrastructure and Fig. 5.

- Pros:
  - Straightforward
  - Guarantees a predictable lifecycle, security, scalability, high availability, and deployment experience across clouds
- Cons:
  - PaaS must be connected to a restricted arrangement of applications and stacks.
  - The reflection is restricted in that it doesn't uncover propelled cloud elements, for example, new networking capabilities.

E. ORCHESTRATION AND AUTOMATION

Orchestration Arrangement is essentially an approach to show the manual procedure required to oversee and send applications and automate them. Automation can be connected to an extensive variety of uses with no change and Fig. 6.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

There are two main families of orchestration platforms.

1. pure-play orchestration
2. container orchestration

- Pros:
  - Can be applied to wide range of applications including legacy, big data, networking, and more complex application stacks.
  - Includes portability of auto-scaling, failover, and continuous deployment processes.
  - Uncovered the full capacities and force of each of the basic clouds services and APIs.
- Cons:
  - Requires adjustment to ensure full portability. Modelling can be a long and at times process.

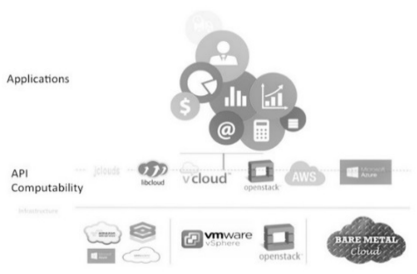


Fig. 4: API Compatibility

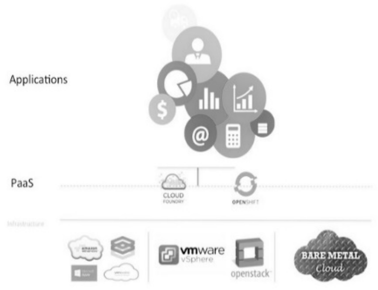


Fig. 5: PaaS

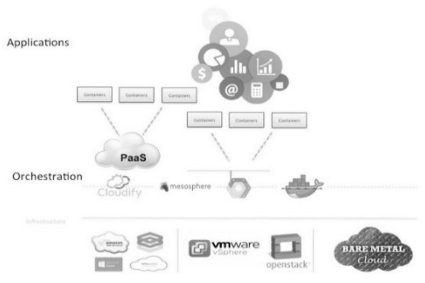


Fig. 6: Orchestration and Automation

III. CHOOSING THE RIGHT COMBINATION

I might want to propose Technically every one of them can be joined together to outwit all worlds. The main question is, what is ideal blend that would yield the best result as far as in terms of portability?
This layered approach empowers the movability of the vastest scope of applications. It guarantees movability of the application double, as well as the transportability of the application management and maintenance

A. ADDING TOSCA TO THE MIX

The approach portrayed above gives a genuinely decent answer for guaranteeing application portability across clouds. But what about the orchestration service? As it were, the portability challenge has been hurled from the cloud infrastructure layer to the orchestration layer.
This is the place TOSCA [10] gets to be convenient, as it gives a standard demonstrating dialect that works crosswise over different cloud infrastructures and Fig. 7.

V. THE TOSCA BUILDING BLOCKS

The OASIS standard Topology and Orchestration Specification for Cloud Applications (TOSCA) were once developed to create portable descriptions for automated management and provisioning of cloud applications. The TOSCA Building Blocks are following

- Application Topologies
- Workflows
- policies

A. Application Topologies

Part in the topology is called Nodes. Every Node has a Type. The Type is unique and hence portable. The Type characterizes Properties and Interfaces. An Interface is an arrangement of snares (named Operations). Nodes are associated with each other utilizing Relationships.

- 3 layers
  - Infrastructure (Cloud or DC objects)

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

- Platform or Middleware (App containers) – Application modules, schemas, and configurations
- Relationships between components:
  - What’s facilitated on what or introduced on what
  - What's associated with what

B. WORKFLOWS

Workflows are automation approach calculations. They delineate the flood of the automation by figuring out which tasks will be executed and when. A task might be an operation, yet it may in like manner be distinctive exercises, including discretionary code.

C. POLICIES

The process TOSCA easy Profile plans to take for policy description with TOSCA service Templates. Moreover, it explores how current TOSCA policy forms and definitions might be applied eventually to express operational policy use cases.

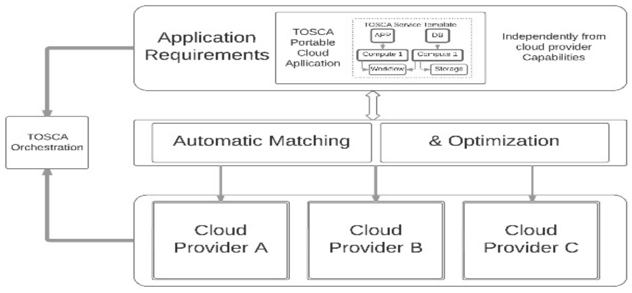


Fig. 7: Adding TOSCA to the Mix

TABLE I: Comparison Between Different Versions

Blocks	TOSCA 1.0	Cloudify 2	Cloudify 3
Workflows	BPMN 2	No change	Raidal
Policies	didn’t elaborate much	DSL on SLA	YAML

IV. CLOUDIFY ARCHITECTURE

Cloudify [9] is based on TOCSA with advance Architecture. It accompanying essential constituents are:

- Command Line Interface client
- Cloudify Director (or) Manager
- Cloudify Agents

A. THE COMMAND LINE INTERFACE CLIENT

The Command Line Interface client is an executable (in python language). The main functions are:

Cloudify Director Bootstrapping - This is, obviously, a discretionary usefulness as you may introduce the supervisor with your Cloudify director tools.

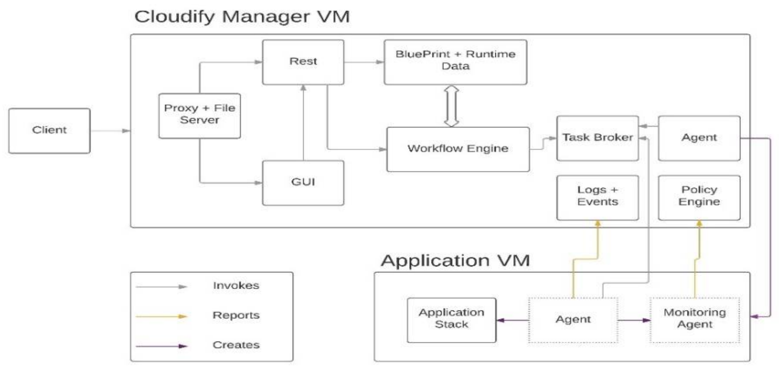


Fig. 8: Manager

Controlling Applications - The Representational State Transfer client interacts the Cloudify director Representational State Transfer interface. It gives the client the full arrangement of capacities for deploying, controlling, and monitoring applications including log and event perusing.

B. THE CLOUDIFY DIRECTOR (OR) MANAGER

The Cloudify Director will deploy and monitor applications described in topology and orchestration plans called blueprints. The manager's fundamental obligation is to run automation forms depicted in work process scripts and predicament execution instructions to the cloudify director agents. The Cloudify director streams and components are talked about in element beneath.

C. THE CLOUDIFY AGENTS

The Cloudify Agents oversee dealing with the administrator's command execution utilizing an arrangement of modules. There's a director side agent each application association and optional operator on each application node virtual machine. Form the Fig. 8 agents are alternatively located on application node virtual machine. This will help for install plugins and execute tasks locally.

VII. **CLOUDIFY MANAGER COMPONENTS**

A. NGINX

Cloudify director utilizes Nginx http as its frontend turn around proxy and file server.

B. REPRESENTATIONAL STATE TRANSFER(REST) API

Cloudify director is controlled by method for a Representational State Transfer API. The Representational State Transfer API covers all the Cloud Orchestration and Management limits.

Cloudify director REST controllers are composed in python language utilizing flask framework and keep running behind a Gunicorn.

C. WEB GRAPHICAL USER INTERFACE

Cloudify director Web Graphical User Interface works versus the REST API however includes extra esteem and perceivability. The GUI has the accompanying screen in Fig. 9.

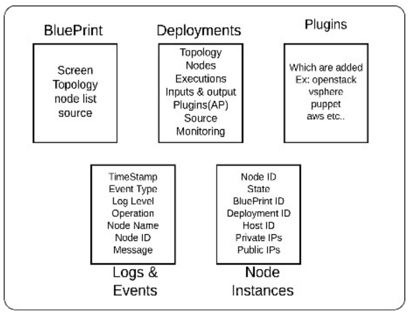

BluePrint
Deployments
Plugins
Screen Topology node list source
Topology Nodes Executions Inputs & output Plugins(AP) Source Monitoring
Which are added Ex: openstack vsphere puppet aws etc..
TimeStamp Event Type Log Level Operation Node Name Node ID Message
Node ID State BluePrint ID Deployment ID Host ID Private IPs Public IPs
Logs & Events
Node Instances

Fig. 9: Web GUI

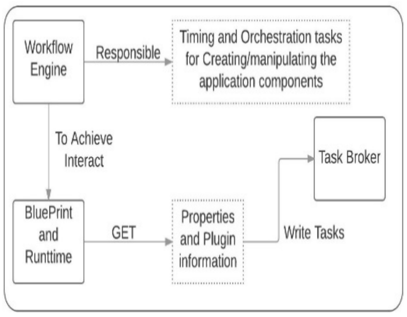

Workflow Engine
Responsible
Timing and Orchestration tasks for Creating/manipulating the application components
To Achieve Interact
Task Broker
BluePrint and Runttime
GET
Properties and Plugin information
Write Tasks

Fig. 10: Workflow Engine

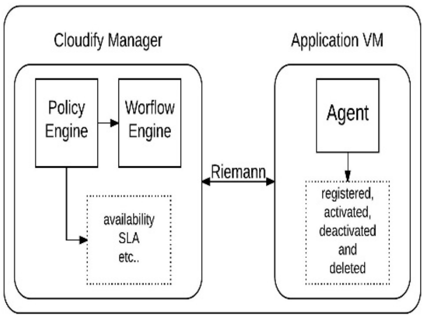


Fig. 11: Policy Engine

D. WORKFLOW ENGINE

Cloudify director utilizes a Workflow engine to take into consideration any automation procedure through inherent and custom workflows. This is built on top of Celery tasks broker [7].

E. RUNTIME ELASTIC

Cloudify director uses Elastic search as its information store for arrangement state. The deployment model and runtime information are put away as JSON records.

F. POLICY ENGINE

Cloudify director offers a policy engine that runs custom policies and Process of policy engine is shown in Fig. 11 Cloudify utilizes Riemann.IO CEP as the centre of the segment.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

G. TASK BROKER

Cloudify director uses Celery with a RabbitMQ message transport to oversee task appropriation and execution. Task Broker can have primary node knowledge of blueprint and runtime properties and this task and operation will be carried out when plugin need to execute.

H. AGENTS

Cloudify operators are loped on Celery daemons specialists and execution as appeared in Fig. 12. An agent can be found remote to the Node it controls or assembled on a similar host.

I. PLUGINS

This are python exteriors for any 1/3 party gadget you must use with any Cloudify work process and execution as appeared in Fig. 13.

J. LOGS AND EVENTS

For troubleshooting and tracing tools Cloudify offers logs and event:

- Events **-** Cloudify reports for every task and execution and it's JSON format.
- Logs **-** Cloudify has a lumberjack that improves log entries with immeasurably imperative environment information.

Log and Events can be shipping, indexing, and storage this is presently for cloudify director in future it is usually extended to help in application to.

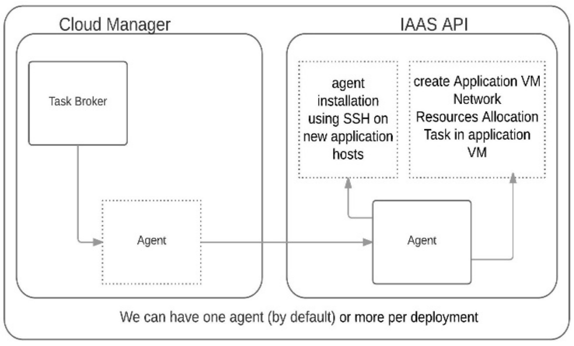

Cloud Manager
IAAS API
Task Broker
agent installation using SSH on new application hosts
create Application VM Network Resources Allocation Task in application VM
Agent
Agent
We can have one agent (by default) or more per deployment

Fig. 12: Agent

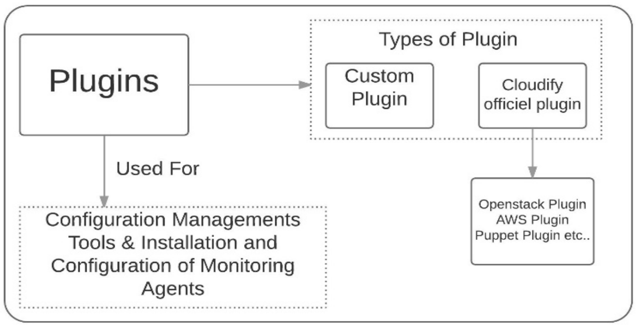


Fig. 13: Plugins

VIII. **CLOUDIFY TECHNICAL FLOWS**

A. BOOTSTRAP

Bootstrapping is the path toward of introducing the Cloudify director. It's finished by utilizing using cfy command and execution as shown in Fig. 14.This diagram depicts the default usage of the bootstrap technique. Since Cloudify Management Environment is communicatedas a blueprint, it can be developed contrastingly be the client. IaaS is a specific case of an environment. A client can choose to bootstrap, for example, on bare metal server.

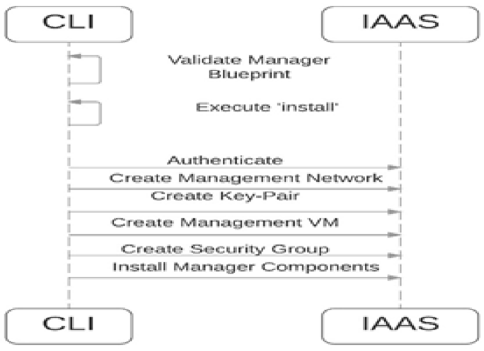


Fig. 14: Bootstrap

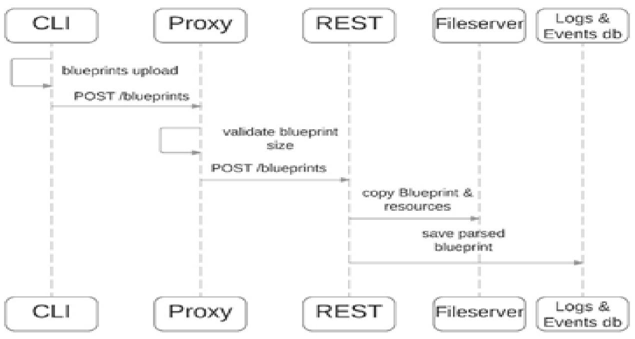


Fig. 15: Upload Blueprint

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Website: www.ijircce.com

Vol. 5, Issue 1, January 2017

B. UPLOAD BLUEPRINT

The underlying stride the supporter must take to put in an application is to have the application Topology and orchestration plan and its related Properties and assets need to transfer to the cloudify director this execution procedure appeared in Fig. 15.

C. DEPLOYMENT CREATION

With an unmistakable complete reason to deploy and control an application we should make a runtime data model within the manager and it continues the report of application state. Diverse arrangements can be created from a solitary blueprint, all things considered once a deployment is made it's free of various even though it is the identical blueprint. The execution of Deployment Creation is shown in Fig. 16.

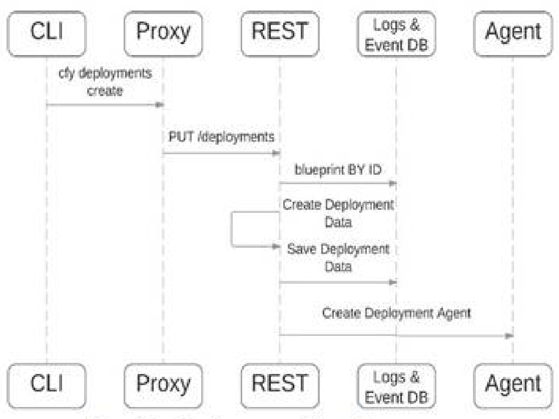


Fig. 16: Deployment Creation

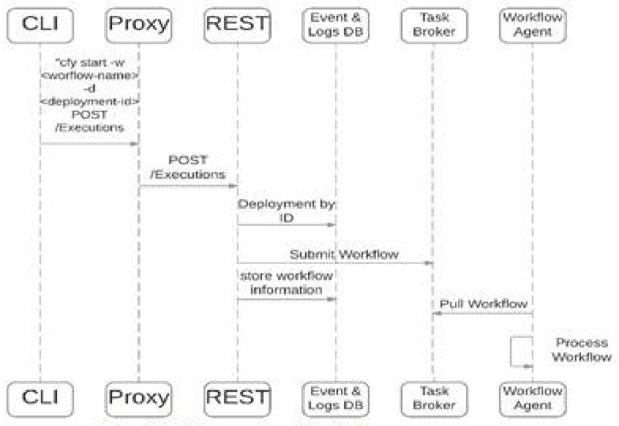
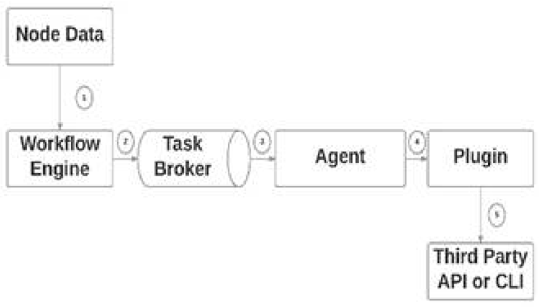


Fig. 17: Execution Workflow

D. WORKFLOW EXECUTION

All automation technique from introductory setup to auto-scaling is done by means of walking a workflow script it is point by point clarified in Fig. 17.

To execute a workflow, utilize the GUI or the CLI cmd. This will make an execution protest for the sending in the manager and run the script. A general block diagram of a workflow execution Fig. 18.



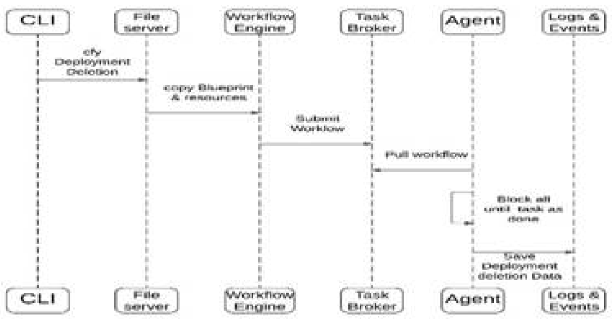


Fig. 18: workflow

Fig. 19: Deployment Deletion

E. DEPLOYMENT DELETION

Utilize the GUI or the CLI where can delete deployments its resources and flow of execution can be seen in Fig. 19. Now, it will erase all data related to the deployments, execution, nodes, and node-instances which are stored in the manager.

V. Conclusion and Future Work

The commitments made in this paper exhibit that a working course of action was found and that TOSCA can be used to fulfil portability to a specific degree. For the future, we expect the importance of an execution space to end up some segment of portability standards, for instance, TOSCA. This allows every Cloud administrator to conform the standard and offer interfaces expected that would pass on Cloud applications displayed with TOSCA
General we can comment that the TOSCA assurance is a basic movement that is overseeing exhaustively with Cloud application portability and orchestration

References

[1] O. T. TC. (2016, Nov.) Topology and orchestration specification for cloud applications version 1.0. [Online]. Available: http://docs.oasisopen.org/tosca/TOSCA/v1.0/csprd01/TOSCA-v1.0-csprd01.pdf
[2] G. Juve and E. Deelman, Automating application deployment in infrastructure clouds, in Cloud Computing Technology and Science (Cloud-Com), 2011 IEEE Third International Conference on, 2011, pp. 658665.
[3] O. C. T. Members. (2016, Aug.) Cloud application management for platforms, version 1.0. [Online]. Available: https://www.oasis-open.org/committees/download.php/47278/CAMP-v1.0.pdf
[4] OpenTOSCA. (2016) Opentosca initiative. [Online]. Available: http: //www.iaas.uni-stuttgart.de/OpenTOSCA/
[5] OpenTOSCA. (2016) Opentosca ecosystem. [Online]. Available: http://les.opentosca.de/v1/
[6] OpenStack. (2016, Oct) Heat - OpenStack Orchestration. [Online]. Available: https://wiki.openstack.org/wiki/Heat
[7] Celery tasks broker [Online]. Available: http://www.celeryproject.org/
[8] Aparna Vijaya, Pritam Dash, and Neelanarayanan V., Migration of Enterprise Software Applications to Multiple Clouds: A Feature Based Approach, Lecture Notes on Software Engineering, Vol. 3, No. 2, May 2015
[9] Cloudify. [Online]. Available: http://getcloudify.org/
[10] TOSCA. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=tosca

BIOGRAPHY

Neelanarayanan Venkataraman is anAssociate Professor in the School of Computer Science Engineering, VIT University. He received PhD in Computer Science, 2007 from IT University of Copenhagen, Denmark. His research interests areDistributed Computing, Cloud Computing, Grid Computing, Network Management and Security, Context - Aware Computing.

Naren Chandra Kattla is a PG Student in the School of Computer Science Engineering, VIT University, currently pursuing their final year of MTech. His research interests are Cloud Computing, Artificial Intelligence.