

Declarative Policy Support for Cloud Application Orchestration

Kena Alexander*, Choonwha Lee**, and Seungmok Chai*

*Division of Computer Science and Engineering, Hanyang University, Seoul, Republic of Korea
 {kenaxle, lee, blaine}@hanyang.ac.kr

Abstract— The orchestration of applications and components in the Cloud is a topic that has been receiving notable attention. As the use and number of cloud computing platforms increases, the need for a standardized means of describing, deploying and managing applications across heterogeneous providers becomes even more apparent. To meet this need we present an extended CAMP platform, capable of describing, deploying and managing applications and components across heterogeneous clouds. Our platform makes use of policies that are described in a declarative YAML format. In order to process policies defined in our extended platform we also present declarative policy directives that specify actions that may be taken in response to our policies. Our extensions to CAMP will allow CAMP compliant platforms to be capable of performing orchestration through the use of declarative policy descriptions, automatically orchestrating applications and components across heterogeneous cloud providers.

Keywords—OASIS CAMP, Orchestration, Policies, Policy Management.

I. INTRODUCTION

Cloud Computing, an ongoing area of Distributed Computing promises the delivery of applications as services over the Internet, as well as hardware and software in the Datacenters that provide those services. The advent of cloud computing promised to provide “users” the benefits of, among many,

- available on demand services
- elimination of up-front commitment
- pay per use model

These benefits, however, come with the addition of inherent issues such as Availability of service, Performance unpredictability, Scaling and Vendor lock-in[1].

The Orchestration of applications and components in across cloud providers is capable of addressing some of the inherent issues within Cloud Computing however the Orchestration of Applications and components in itself is not a simple task.[2] The Orchestration of applications and resources in the cloud involves dynamically deploying, managing and maintaining those aforementioned components in and across multiple heterogeneous cloud platforms in order to meet the needs of clients. As it is possible for cloud providers to adopt combination of a number of technologies and API's [3] it is can be seen that standardization ([4], [5]) can provide the answer to

Orchestration of applications and components across heterogeneous cloud platforms.

The current standard for cloud application description, deployment and management, OASIS CAMP (Cloud Application Management for Platforms)[6], provide a means of declaratively interacting with cloud applications and providers. However, OASIS CAMP is unable to provide orchestration features that will allow it to deploy and manage application components across heterogeneous cloud platforms.

In this paper, we aim to present our proposed extension to CAMP which will allow for declarative policy management and orchestration of deployed application components. Through this we argue that it will be possible to declaratively orchestrate applications and components in CAMP through a declarative policy specification.

II. CAMP

CAMP represents an industry accepted, emerging standard by OASIS, defined to tackle the problems of cloud application definition, deployment and management. OASIS CAMP provides a standardized means for deploying, monitoring and managing applications by allowing the persons to describe the format of as well as how application components should be deployed to CAMP compliant providers using YAML plans. A CAMP plan written in YAML may contain a list of artifacts to be deployed to a provider, the services the provider may use to fulfill the installation of the artifacts and the relationship between each artifact and its required service. CAMP however does not provide the facilities needed to orchestrate an application or components across multiple heterogeneous cloud. An application and its components presented to a CAMP platform as a deployment plan is deployed to and managed on a single provider.

III. DECLARATIVE POLICIES

In this section, we present the overall strategy of our declarative policy specification as well as the architecture behind our approach.

A. Overall strategy

The overall approach of our work focuses on providing extensions to CAMP allowing for declarative policy specification and management. For this we aimed to

- Include declarative policies within CAMP plans

- Provide a declarative mechanism for processing policies.

1) Architecture

The architecture of our approach (Figure 2) consists including a policy and policy management components to CAMP's architecture of a TOSCA Parser and an extended CAMP platform joined by a conversion engine. Tosca documents inclusive of policies are converted into our extended CAMP format for deployment.

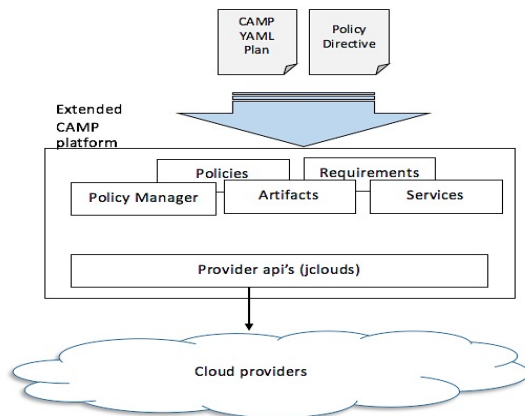


Figure 1. This depicts the architecture of the extendedCamp solution

2) Policy specification

Our extended CAMP platform specifies policies that can be used to manage a deployed application or components. Our approach introduces declaratively defined, typed policies within our extended CAMP document making use of two main components, typed policies and typed constraints

3) Typed Policies and Constraints

Declaratively defined policies in our extended CAMP platform (Figure 2) specify state or behavior that an entity should adhere to and does not imperatively specify actions that must be taken. Types can be used to categorize policies based on their mode of operation. Typed constraints capture the state that an entity should ideally be in if it is to conform to the policy. For example, a Placement policy may be associated with an entity stating the entity should be started (**SERVICE_UP = true**) and its location should ideally be chosen from one of a supplied set of locations (**PROVISIONING_LOCATION is within [loc1, loc2, ..., locx]**).

```

policies:
- type: kr.ac.hanyang.oCamp.entities.policies.Placement
  constraints:
  - property: SERVICE_UP
    type: kr.ac.hanyang.oCamp.entities.constraints.EqualTo
    value: true
  - property: PROVISIONING_LOCATION
    type: kr.ac.hanyang.oCamp.entities.constraints.Within_Affinity
    value: [AWS Virginia (us-east-1), AWS Tokyo (ap-northeast-1)]
    targets: [webapp_cluster]

```

Figure 2. Policy specification in our extended CAMP platform.

4) Policy management

As Declaratively defined policies define criteria that an entity should adhere, whether or not the entity adheres depends not on the policy but the policy enforcer ([7], [10]). Policy managers were therefore introduced into our extended CAMP platform as an entity that can interpret and enforce a policy on an entity. The role of the policy manager is to determine the best action to enforce a policy. This is done through the use of policy management directives.

The policy management directives (Figure 3) are defined using as a group of actions that the policy manager can choose from that are applicable to the type of entity that the policy is associated with. Each action identifies the property of the entity that this action influences and the transition the property will undergo in the event of the action being performed. For example, a placement policy manager may be given the directives that specify two possible course of actions it may take, **START** or **STOP**. The **START** action operates on an entity by setting its **SERVICE_UP** property to true from its initial value of false. The same action also operates on the entity's provisioned location by setting its **PROVISIONING_LOCATION** value to "any" specified value from its initial value of nothing (null).

```

type: kr.ac.hanyang.oCamp.entities.policies.Placement
actiongroups:
- id: START
  actions:
  - property: SERVICE_UP
    transitions:
  - type: Initial
    value: false
  - type: Set
    value: true
  - property: PROVISIONING_LOCATION
    transitions:
  - type: Initial
    value: null
  - type: Set
    value: ANYTHING
- id: STOP
  actions:
  - property: SERVICE_UP
    transitions:
  - type: Initial
    value: true
  - type: Set
    value: false
  - property: PROVISIONING_LOCATION
    transitions:
  - type: Initial
    value: ANYTHING
  - type: Set
    value: null

```

Figure 3. Policy management directive

As these actions represent known actions, that can be invoked on entities within our extended CAMP platform, the policy manager needs only to decide on the best course of action to be performed and invoke that action on the entity. The entity should already be aware of how to carry out that action thus imperative instructions are not passed to the entity.

IV. EVALUATION

In order to demonstrate the capabilities of our approach we created a used an extended CAMP plan consisting of a web cluster and database server as well as placement and management policies for placing the components of the application in different clouds. This was then deployed and managed using our extended CAMP platform. We also defined management policies for the web cluster defining different loads and recorded the responses of the policy managers to each situation. The responses by our extended CAMP platform are recorded in Table 1 and Table 2.

TABLE 1. WEB CLUSTER POLICY MANAGEMENT RESPONSES

Policy Type	Desired State	Actual State	Action Chosen
Placement	State: STARTED Location: (AWS Virginia or AWS Tokyo)	State: UNSTARTED Location: (null)	State: START Location: (AWS Tokyo)
QoS	Nodes: 2 5 < Req/Sec < 10	Nodes: 2 Req/Sec > 15	SCALE_UP Nodes: 3 Req/Sec = 5
QoS	Nodes: 2 5 < Req/Sec < 10	Nodes: 3 Req/Sec < 3	SCALE_DOWN Nodes = 2 Req/Sec = 6

TABLE 2. SQL SERVER POLICY MANAGEMENT RESPONSES

Policy Type	Desired State	Actual State	Action Chosen
Placement	State: STARTED Location: (Rackspace-HKG or Rackspace-SYD)	State: UNSTARTED Location = (null)	State: START Location: (Rackspace-HKG)

V. CONCLUSION

Our work is motivated based on the fact that Cloud Application Orchestration can solve problems with Cloud Computing however in order to ensure that inherent problems are not compounded, a standards based approach should be utilized. We presented CAMP as a standard focused on Application deployment and management. We also presented that through extensions to CAMP it is possible, through policies, to provide a standard, orchestration-capable, application delivery platform. Finally, showed that our extended CAMP platform is capable of automatically orchestrating applications and components across heterogeneous cloud providers.

ACKNOWLEDGEMENT

This research was supported by the MSIP (Ministry of Science, ICT & Future Planning), Korea, under the ITRC(Information Technology Research Center) support program (IITP-2016-H8501-16-1006) supervised by the IITP(Institute for Information & communications Technology Promotion) and under the National Program for Excellence in SW (IITP-2016-R7719-16-1003) supervised by the

IITP(Institute for Information & communications Technology Promotion).

REFERENCES

- [1] M. Armbrust, A. Fox, R. Griffith, A. Joseph, and R. H., "Above the clouds: A Berkeley view of cloud computing," *Univ. California, Berkeley, Tech. Rep. UCB*, pp. 07–013, 2009.
- [2] C. Liu, B. T. Loo, and Y. Mao, "Declarative automated cloud resource orchestration," p. 26:1–26:8, 2011.
- [3] N. Ferry, A. Rossini, F. Chauvel, B. Morin, and A. Solberg, "Towards Model-Driven Provisioning, Deployment, Monitoring, and Adaptation of Multi-cloud Systems," *2013 IEEE Sixth Int. Conf. Cloud Comput.*, pp. 887–894, 2013.
- [4] A. Barros, M. Dumas, and P. Oaks, "Standards for web service choreography and orchestration: Status and perspectives," *Bus. Process Manag. Work. LNCS 3812*, pp. 61–74, 2006.
- [5] W. Bumpus, "NIST Cloud Computing Standards Roadmap," *NIST Cloud Comput. Stand.*, pp. 1–3, 2013.
- [6] P. (Ca T. Lipton, S. (Ibm) Moser, D. (Vnomic) Palma, and T. (Ibm) Spatzier, "Topology and Orchestration Specification for Cloud Applications," no. November, pp. 1–114, 2013.
- [7] F. Limited and R. Hosting, "Cloud Application Management for Platforms Version 1.1 Committee Specification Draft 04 /," vol. 2014, no. February, 2014.
- [8] W. Draft, "TOSCA Simple Profile in YAML Version 1.0," no. March, pp. 1–83, 2014.
- [9] L. Kagal, T. Finin, and A. Joshi, "Declarative Policies for Describing Web Service Capabilities and Constraints," *W3C Work. Constraints Capab. Web Serv. Oracle Conf.*, 2004.
- [10] C. Dimoulas, S. Moore, A. Askarov, and S. Chong, "Declarative policies for capability control," *Proc. Comput. Secur. Found. Work.*, vol. 2014–Janua, pp. 3–17, 2014.



Kena Alexander is a M.S. student at Hanyang University, Seoul, South Korea. He received his B.S. in Computer Science from The University of the West Indies, St Augustine, Trinidad and Tobago in 2005. His research areas are cloud application orchestration and inter-cloud portability and management.



networking and computing, and services computing technology.

Choonghwa Lee is a professor in the Division of Computer Science and Engineering at Hanyang University, Seoul, South Korea. He received his B.S. and M.S. degrees in computer engineering from Seoul National University, South Korea, in 1990 and 1992, respectively, and his Ph.D. degree in computer engineering from the University of Florida, Gainesville, in 2003. His research interests include cloud computing, peer-to-peer and mobile



Seungmok Chai is a student at Hanyang University, Seoul, South Korea. His research interests include cloud and distributed computing systems and computer networking protocols.