Multi-Tier Autonomic Resource Orchestration Framework for Dynamic Service Elasticity across Federated Cloud Environments

Nitin Sikri

Department of Electrical Engineering,
IET, MJP Rohilkhand University,
Bareilly, Uttar Pradesh, India
nitinsikri23@gmail.com

Abstract. The Cloud-SAP framework employs a layered autonomic computing strategy to retain Quality of Service (QoS) of tenant applications in heterogeneous and distributed cloud environments.
Cloud computing uses a layered architecture composed of five levels: first, execution environment controllers (EAC); second, containerised workload orchestrators (CWO); third, application stack managers (AS); fourth, cloud instance provisioners (CP); and fifth, "federated cloud" (FC) manager for negotiating between providers.
Adaptive scaling through horizontal and vertical provisioning is facilitated by feedback control loops at all levels. A proof-of-concept implementation on OpenNebula and extensive experiments and simulations validate the approach. The results indicate that the solution reduces the cost of deployment by 20% as compared to single-layer scaling solutions, while enabling consistent application performance through coordinated decisions made across abstraction layers.
The architecture is ideal for hybrid and federated clouds with efficient collaboration of multiple providers for dynamic workloads. This approach offers an agnostic federation model for providers and a reference architecture that can be reused for enterprise cloud systems with complex multi-layered resource management. . . .

Keywords: Autonomic computing, cloud federation, service elasticity, resource orchestration, quality of service, self-adaptive systems, cloud bursting, multi-tier architecture.

1 Introduction

Cloud computing enables on-demand, scalable resources with pay-as-you-go models [17], creating expectations of high availability and reliability. However, maintaining consistent Quality-of-Service (QoS) remains challenging due to dynamic workloads.

Traditional systems scale resources at isolated layers rather than across the full stack. PaaS platforms offer limited auto-scaling based on simple thresholds

M. Kaushik et al. (eds.), *Proceedings of the International Conference on Responsible, Risk-aware, and Regulated AI (RRRAI 2026)*, Advances in Intelligent Systems Research 210,
https://doi.org/10.2991/978-94-6239-723-1_31

[18]. For example, Amazon EC2 Auto Scaling and OpenShift rely on request-based scaling, while Heroku requires manual intervention [19]. These approaches are inadequate for enterprise environments requiring multi-provider coordination and fine-grained control.

Prior work addresses these issues partially. InterCloud [20] introduces federation but lacks self-adaptation, while Carina [29], OneFlow [30], and Cloud-Foundry [31] support scalability but operate at single abstraction levels without full autonomic capabilities.

This paper proposes Cloud-SAP, inspired by IBM autonomic computing [21], featuring a five-layer architecture enabling self-configuration, optimization, healing, and protection. Contributions include: (1) a multi-layer autonomic resource management architecture, (2) a proof-of-concept implementation, and (3) evaluation showing improvements in cost, utilization, and QoS.

The remainder of the paper is organized as follows: Section **??** covers background, Section 3 presents the architecture, Section 4 details implementation, Section 5 discusses results, and Section 6 concludes.

2 Related Work

Machine learning has improved cloud resource management via optimization techniques such as Bayesian tuning [4] and learning-based calibration [1].

Handling uncertainty is critical in federated clouds. Adaptive reweighting [5] improves prediction reliability, while multi-task learning [8] optimises cost, performance, and reliability.

Anticipatory autonomic management [12] supports proactive scaling and failure prediction, while spatial and agent-based models [10] assist distributed infrastructure management.

Efficiency is enhanced through tensor optimisation [14] and continual learning [13]. Infrastructure challenges [2] highlight the need for resilient systems, and energy disaggregation [3] improves monitoring granularity.

NLP techniques [7, **?**] support policy management, while sentiment analysis [11], dataset construction [16], and summarisation [15] aid evaluation and log analysis. Health modelling [9] further supports QoS monitoring under uncertainty.

Cloud-SAP integrates these approaches into a unified hierarchical framework, enabling federation-aware, multi-layer autonomic management.

3 Cloud-SAP Architecture

3.1 Architectural Overview

Based on IBM's autonomic computing model, the Cloud-SAP comprises five layers, with each layer controlled by autonomic controllers. This allows coordinated management across the cloud stack. The architecture and relationships between resources and managers are shown in Figure 1.

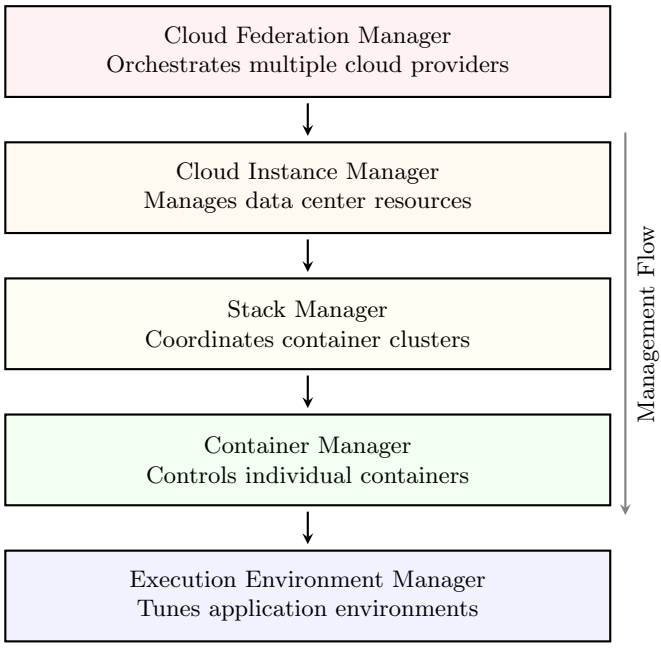

Fig. 1. Cloud-SAP Hierarchical Architecture

Every layer applies a complete MAPE loop and connects its sensors and effectors to adjacent layers. Using orchestrating managers, this hierarchy allows for local optimisation with global coordination while being provider agnostic via standard interfaces.

3.2 Managed Resources Hierarchy

Cloud-SAP defines six resource abstraction levels:

1. **Application Execution Environment**: Runtime environments (e.g., JVM, .NET CLR, Python) requiring tuning.
2. **Container**: Isolated units (VMs, containers) ensuring resource boundaries.
3. **Stack**: Homogeneous container clusters (e.g., web or database groups).
4. **Service**: Composite applications of multiple stacks.
5. **Cloud Instance**: Provider-specific infrastructure managing services.
6. **Cloud Federation**: Interconnected cloud instances via standard protocols.

The performance of the lower-layer affects the quality of service at the higher-layer. Hierarchical autonomic control manages these dependencies.

3.3 Autonomic Manager Design

Each manager implements a MAPE loop with layer-specific adaptations:

Monitoring: Uses provider-specific sensors with aggregation and adaptive intervals based on volatility and policies.

Analysis: Applies statistical and machine learning models, supporting parallel execution based on QoS needs.

Planning: Generates actions considering dependencies, priorities, and cooldown constraints.

Execution: Enforces actions via consistent effectors to ensure stable reconfiguration.

Figure 2. The MAPE-based autonomic manager is illustrated as the components utilise knowledge and learn from history for adaptation.

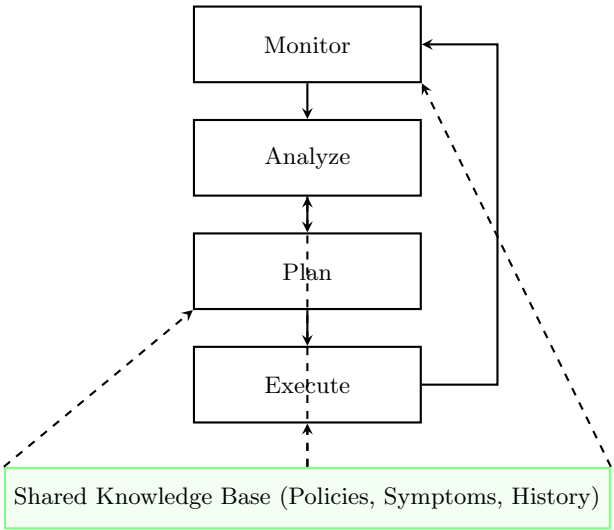

Fig. 2. Autonomic Manager Control Loop with Knowledge Sharing

3.4 Federation Management

The cloud federation manager of Cloud-SAP deploys resources from several cloud suppliers with two specialised autonomic managers.

1. **Monitor and Analyse Manager**: Continuously evaluates availability, cost, and performance of federated resources, forecasts demand, and identifies scaling opportunities.

2. **Plan and Execute Manager**: Through analysis of insight, it uses multi-criteria optimisation of cost, performance, and reliability to determine deployment strategies across providers.

The federation manager uses market-based resource allocation models [28], where it treats a cloud instance as an economic entity to allow dynamic pricing and the efficient distribution of the resources.

4 Proof-of-Concept Implementation

4.1 Implementation Overview

The implementation of Cloud-SAP proof of concept was developed with four autonomic managers including container manager, stack manager, cloud instance manager, and cloud federation manager. The design exhibits feasibility and remains extensible for future components. The architecture for system deployment is illustrated in figure 3.

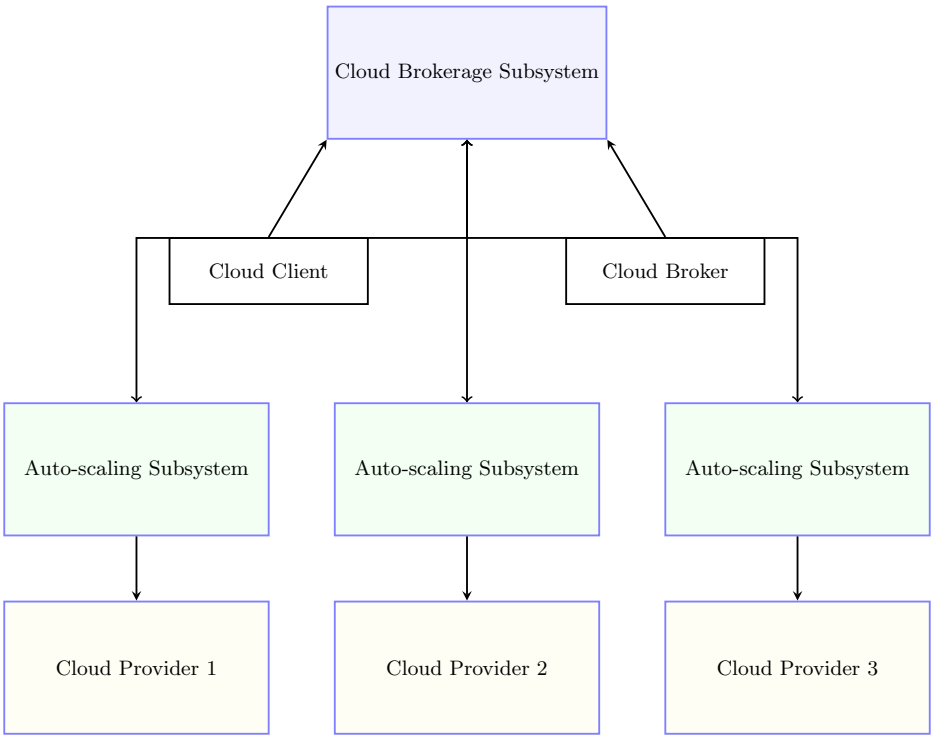

Fig. 3. Cloud-SAP Implementation Architecture

The OpenNebula cloud manager is an extensible platform supporting multiple hypervisors. OpenVZ container virtualisation has been adopted owing to its light-weight and dynamic adjustment of resources. Ruby portions allow fast prototyping.

4.2 Component Implementation Details

Container Manager: Utilises OpenVZ user bean counters for vertically scaling CPU in 30% increments. It uses threshold-based analysis with customizable bounds and will delegate to stack managers when local resources run out.

Stack Manager: Integrates container clusters using OpenNebula API. It uses Appflow templates to ensure horizontal scaling, load balancing, and reliable monitoring of cluster health.

Cloud Instance Manager: Functions as a supplier substitute, revealing capacity and pricing towards federation managers. It schedules basic tasks and takes part in resource negotiations.

Cloud Federation Manager: Uses asynchronous messaging based on AMQP, connects clients and providers in a cost-effective manner using greedy algorithms based on the pricing per stack.

4.3 Policy Specification Framework

Cloud-SAP uses JSON-based policies supporting:

- **Threshold policies**: Limits for CPU, memory, and response time
- **Composite policies**: Logical combination of conditions
- **Temporal policies**: Time-based workload handling
- **Cost policies**: Budget and optimization constraints

Policies are evaluated at configurable intervals with cooldowns to prevent oscillation and can be inherited or overridden across resource levels.

5 Experimental Evaluation

5.1 Evaluation Methodology

Cloud-SAP is compared to Carina [29]. Performance index is evaluated along three dimensions: (1) minimisation of deployment cost in multi-provider setups; (2) auto-scaling in single-provider setups; and (3) cloud bursting in federations.

Different hardware configurations were used for experiments to simulate real datacenters. Resource monitoring utilised simulated services to facilitate repeatable workload generation through polynomial interpolation. Experiments were conducted 10 times and averaged.

5.2 Deployment Cost Optimization

This experiment investigated the reduction of cost by selecting optimum providers. We tested three providers with different pricing models against five stack types: Java, Ruby, PostgreSQL, Python and AMQP. In Table 1, price matrices and optimal allocations are shown.

Cloud-SAP successfully attained an overall deployment cost of $1080 with an optimal distribution of stacks. This represents a 16.9 saving against the best alternative single-provider scenario. Resource allocation that is aware of federation can result in economic benefits, especially to heterogeneous applications.

Table 1. Deployment Cost Analysis and Provider Selection

Stack Type	Provider 1	Provider 2	Provider 3	Selected
Java	$150	$120	$180	Provider 2
Ruby	$220	$290	$250	Provider 1
PostgreSQL	$320	$240	$290	Provider 2
Python	$200	$260	$180	Provider 3
AMQP	$330	$390	$285	Provider 3
Total Cost	$1220	$1300	$1185	$1080
Savings	11.5%	16.9%	8.9%	-

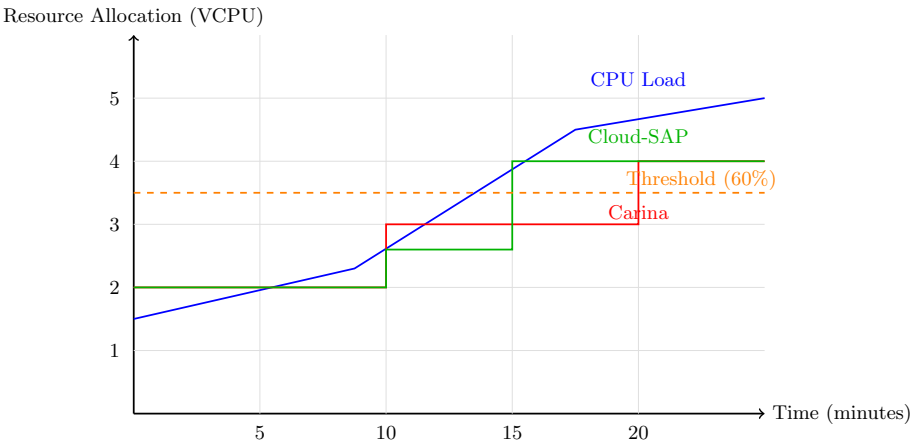

Fig. 4. Resource Allocation Comparison During Auto-Scaling

5.3 Single-Provider Auto-Scaling Efficiency

The second experiment examined efficiency in terms of computing resources. We implemented the same Java application stack in both platforms and then exposed them to synthetic CPU load patterns resembling actual business cycles. Figure 5 resource allocation and cost comparison over time.

According to Cloud-SAP, it opted for vertical scaling for efficient resource use without performance drop and over-provisioning reduction. Carina costs 15.3 currencies according to an analysis; the estimate for Cloud-SAP is 15.19, which provides a convenient estimate of a savings of 0.7%. The estimate could not be more aggressive in terms of performance maintenance. Cloud SAP was capable of executing smoother resource transitions without substantial capacity changes.

5.4 Cloud Bursting Effectiveness

The final experiment tested federation capabilities under resource depletion conditions. We configure two cloud service providers with limited resources, but they

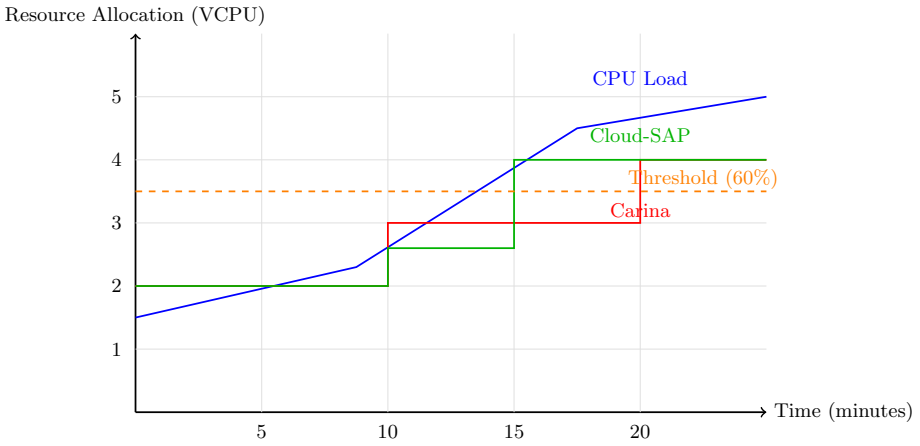

Fig. 5. Resource Allocation Comparison During Auto-Scaling

have a sufficient capacity when combined together. The illustration shows the processing capacity beyond one supplier.

Cloud-SAP attained 283 trans/s by vertical scaling and cloud bursting, while Carina achieved 180 trans/s due to single provider limits. The 57% capacity gain shows the power of federation for dynamic workloads.

6 Conclusion and Future Work

Cloud-SAP employs a hierarchical design with five layers of abstraction to manage multi-provider cloud resources. It supports coordinated scaling up or down, and left or right, while being federation-aware, overcoming limitations of current platforms.

The results of the experiment demonstrate that comparing the costs of multi-provider deployment reduces costs by 16.9%. It improves the utilisation of resources by vertical scaling. It increases capacity by 57% through cloud bursting. These results validate the approaches used for enterprise deployment.

Future work involves improving the prediction accuracy with the use of machine learning models [26], optimising economically using cost-aware policies and dynamic pricing, and using upcoming standards (OCCI, CIMI) for better interoperability. Using enterprise benchmarks such as SPECweb and TPC-C will be examined as well.

The architecture lays the groundwork for next-generation cloud platforms where hierarchical and self-adaptive frameworks are crucial to ensuring QoS and cost savings in hybrid and multi-cloud settings.

References

1. Akhil Veluru. *A Learning-Based Approach for Sensor Calibration in SWOT Sea Surface Height Observations*. 2024. DOI: https://doi.org/10.31224/5008.
2. Akhil Veluru. *Smart Energy System Deployment in Developing Nations: Addressing Infrastructure Challenges*. 2024. DOI: https://doi.org/10.31224/5015.
3. Akhil Veluru. *Energy Disaggregation Using Binary Indicators for Appliance On/Off States*. 2024. DOI: https://doi.org/10.31224/5010.
4. Akhil Veluru. *Bayesian Optimization of Hyperparameters for Rainbow DQN in the CartPole-v1 Environment*. 2024. DOI: https://doi.org/10.31224/5009.
5. Ashutosh Agarwal. *Reducing Variability through Adaptive Weighting for Reliable Predictions: Improving Model Resilience Across Varied Domains*. In *2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS)*, pages 1–5, 2025. DOI: 10.1109/WorldSUAS66815.2025.11198985.
6. Ashutosh Agarwal. *Comparative Analysis of Text-Embedding Models for Enhanced Search Functionality: A Case Study on Harvard Course Data*. In *2025 World Skills Conference on Universal Data Analytics and Sciences (WorldSUAS)*, pages 1–7, 2025. DOI: 10.1109/WorldSUAS66815.2025.11199061.
7. Ashutosh Agarwal. *Optimizing Classification of Infrequent Labels by Reducing Variability in Label Distribution*. 2025. arXiv: https://doi.org/10.48550/arXiv.2511.07459.
8. Ashutosh Agarwal. *Tri Risk: A Multi-Task Deep Learning Framework for Enterprise Cyber Threat Risk Assessment*. Preprints, 2025. DOI: https://doi.org/10.20944/preprints202511.1643.v1.
9. Pramesh Baral. *Unobserved Health: The Impact of Reporting Error on Health Dynamics*. Preprints, 2025. DOI: https://doi.org/10.20944/preprints202512.1756.v1.
10. Pramesh Baral. *Integrating Spatially Continuous Environmental Covariates into HMMs for Animal Behavior via Kriging*. TechRxiv, 2025. DOI: 10.36227/techrxiv.176369836.61018639/v1.
11. Pramesh Baral. *Deep Learning for Text-Based Sentiment Prediction*. TechRxiv, 2025. DOI: 10.36227/techrxiv.176369697.70526974/v2.
12. Pramesh Baral. *Anticipatory Autonomic Management of Poly-Cloud Environments: A Machine Intelligence Paradigm*. TechRxiv, 2025. DOI: 10.36227/techrxiv.176369699.93881867/v1.
13. Mridul Banik. *LLM Tuning: Neural Language Persistence through Adaptive Mixture*. In *Proceedings of the 2024 Conference on Neural Information Processing Systems*, 2024. DOI: https://doi.org/10.1145/3774791.3774803.
14. Mridul Banik. *Novel Tensor Norm Optimization for Neural Network Training Acceleration*. In *Proceedings of the 2024 Conference on Neural Information Processing Systems*, 2024. DOI: https://doi.org/10.1145/3774791.3774805.
15. Mridul Banik. *Instructional Video Summarization with Transformers: A Curriculum Learning Approach for ASR-Generated Transcripts*. 2024. DOI: https://doi.org/10.31224/5662.
16. Mridul Banik, Md. Jamilur Rahim Rifat, Jannatun Nahar, Nafiz Hasan, and Fariha Rahman. *Okkhor: A Synthetic Corpus of Bangla Printed Characters*. In *Proceedings of the Future Technologies Conference (FTC) 2020, Volume 1*, pages 689–703, Springer, Cham, 2021. DOI: https://doi.org/10.1007/978-3-030-63128-4_53.
17. P. Mell and T. Grance, "The NIST definition of cloud computing," National Institute of Standards and Technology, Special Publication 800-145, 2011.

18. L. M. Vaquero, L. Rodero-Merino, and R. Buyya, "Dynamically scaling applications in the cloud," ACM SIGCOMM Computer Communication Review, vol. 41, no. 1, pp. 45-52, 2011.
19. L. Leong et al., "Magic quadrant for cloud infrastructure as a service," Gartner Research, G00246970, Aug. 2013.
20. R. Buyya, R. Ranjan, and R. N. Calheiros, "InterCloud: Utility-oriented federation of cloud computing environments for scaling of application services," in Proc. International Conference on Algorithms and Architectures for Parallel Processing, 2010, pp. 13-31.
21. IBM Corporation, "An architectural blueprint for autonomic computing," IBM White Paper, 4th Edition, 2005.
22. Y. Brun et al., "Engineering self-adaptive systems through feedback loops," in Software Engineering for Self-Adaptive Systems. Springer, 2009, pp. 48-70.
23. M. Litoiu, M. Woodside, and T. Zheng, "Hierarchical model-based autonomic control of software systems," in Proc. International Conference on Autonomic Computing, 2005, pp. 325-326.
24. A. N. Tantawi and D. Towsley, "Optimal static load balancing in distributed computer systems," Journal of the ACM, vol. 32, no. 2, pp. 445-465, 1985.
25. M. Abdeen and M. Woodside, "Seeking optimal policies for adaptive distributed computer systems with multiple controls," in Proc. International Conference on Parallel and Distributed Computing, Applications and Technologies, 2002, pp. 487-492.
26. S. Islam, J. Keung, K. Lee, and A. Liu, "Empirical prediction models for adaptive resource provisioning in the cloud," Future Generation Computer Systems, vol. 28, no. 1, pp. 155-162, 2012.
27. HAProxy Documentation, "Load balancing algorithms and configuration," 2013. [Online]. Available: http://haproxy.lwt.eu/
28. R. Buyya, D. Abramson, J. Giddy, and H. Stockinger, "Economic models for resource management and scheduling in grid computing," Concurrency and Computation: Practice and Experience, vol. 14, no. 13-15, pp. 1507-1542, 2002.
29. Carina Project, "OpenNebula Carina environment manager," 2014. [Online]. Available: https://github.com/blackberry/OpenNebula-Carina
30. OneFlow Documentation, "OpenNebula application flow and auto-scaling," 2014. [Online]. Available: http://docs.opennebula.org
31. Cloud Foundry Documentation, "Cloud Foundry open-source PaaS platform," 2013. [Online]. Available: https://www.cloudfoundry.org
32. OpenShift Documentation, "Red Hat OpenShift PaaS platform," 2013. [Online]. Available: https://www.openshift.com