Proceedings of the Federated Conference on Computer Science and Information Systems pp. 711–715
DOI: 10.15439/2019F82
ISSN 2300-5963 ACSIS, Vol. 18

A Design and Experiment of Automation Management System for Platform as a Service

Alalaa Tashkandi
Saudi Aramco, Information Technology, Al-Midra Tower, Dhahran 31311, Saudi Arabia
Email: alaa.tashkandi@aramco.com

***Abstract*—Security [11] and quality [4] of cloud computing (CC) services represent significant factors that affect the adoption by consumers. Platform as a Service (PaaS) is one of CC service models [14]. Management of database systems, middleware and application runtime environments is automated in PaaS [2]. PaaS automation management issues and requirements were collected in three rounds from information technology experts using Delphi technique. In this paper, PaaS automation quality and security management system (MS) layered model is proposed and validated. The aim of the MS is enabling PaaS model for mission critical platforms. Validation of the MS was based on experiment in a private cloud for an organization undergoing a transformation toward PaaS computing.**

I. Introduction

SERVICE interruption caused by Platform as a Service (PaaS) automation failure or security incidents may lead to reputation damage or reimbursement cost. PaaS management system (MS) should provide integrated proactive and reactive control for quality and security events.

The MS model was achieved based on analysis of experts' inputs using Delphi technique in a Cloud Service Provider (CSP) organization providing mission critical services. The model enforces secure rollout, update and monitoring of automation artifacts. Additionally, quality assurance (QA) process is enforced. After three rounds of communications, verifications and feedbacks, PaaS MS model was finalized. Observations based on MS experiment are summarized in Analysis and Discussion section of this paper.

In this research, we have collected the adoption factors and requirements of PaaS for mission critical platforms to drive a novel management model using Delphi approach. The model was experimented and validated in a private cloud environment.

II. Related Work

Applications in PaaS context can be classified based on computation model, resource utilization type, resource utilization variability or interactivity level. Examples of PaaS offering include Google App Engine and Amazon Web Services Lambda [2]. PaaS is one of three service models in Cloud Computing (CC). In this model, customer does not have control over the infrastructure layer while he has control over the application and its configuration [14]. Middleware, application runtime environment and database systems are examples of components in PaaS model [3], [5], [10], [13]. PaaS provides self-deployment, monitoring and lifecycle management of applications [2], [4]. PaaS offers include middleware services, like Database as a Service (DbaaS), and reusable middleware components. Middleware components include application runtime environments that can be used as part of an application platform [5], [8].

PaaS provides flexibility and agility to developer team. By automating the deployment of application platforms and their components, different setup and configuration can be realized with minimal efforts and time. Interactions between developer team and operation team are minimized through automation [3], [7]. Infrastructure is abstracted by PaaS [5].

Private cloud and multicloud are now the convention of organizations that need to protect their information technology investment while seeking new technologies and opportunities. Use of resources in multicloud can be sequential or parallel. Support of multicloud minimizes vendor lock-in issue associated with the Cloud [2], [7], [12].

Two sets of users represent PaaS consumers. The first set represents software developers in organizations. The other set represents SaaS CSPs who needs to focus on Software development and services quality [7].

A. PaaS Challenges

The use of CC in general provides speed and flexibility to organization. On the other hand, cloud raises challenges in terms of quality and security [4], [11]. When PaaS is evaluated by organization for production and business operation, security and QA are two significant adoption factors. Service Level Agreement (SLA) is one of the solutions that was studied in several papers [2], [16]. Service level quality is measured based on Service Level Objectives (SLOs). Service Level Agreement (SLA) grants SLOs based on contractual agreement between the service provider or a broker and the customer [2]. Sequence of events and logs of security control and QA must be forensically sound and reliable in case of a security incident or PaaS failure [1]. Under PaaS service model, wide varieties of technologies exist [4]. Spe-

This work was sponsored by Computer Operations Department, Aramco.

cific applications in PaaS are based on distributed computing. Managing the dependency between distributed components is complex [10].

B. PaaS Automation

Application deployment automation is enabler for PaaS [6]. Automation performance can be evaluated qualitatively or quantitatively [5]. Operation automation approaches were classified into infrastructure management, plan-based configuration management, image-based configuration management, model-based management and platform centric management automation [9]. Automation of middleware and application deployment should be decoupled from Infrastructure layer. Encapsulating applications in virtual images reduces the flexibility of updating applications over time [6].

Automation standardizations; such as Topology and Orchestration Specification for Cloud Applications (TOSCA); are emerging. TOSCA has relatively small community. Released automation artifacts are limited [9], [15], [17].

III. Data Collection

Standards in PaaS are emerging. Security and quality control processes in this context are under development. In this research project, efforts were devoted to build a MS for PaaS that is suitable for organization mission critical operations.

A. Research Methodology

Delphi approach was followed to collect data and feedback from Information Technology experts within private cloud provider organization. The first round was started after building PaaS automation system without quality and security control in a dedicated computing lab environment. Several Automation scenarios were tested to deploy Database and application servers, and to manage their lifecycle in PaaS context. Feedbacks from IT security team, datacenter infrastructure team and middleware operation team were collected.

Based on the feedback from the experts, the system was customized with control measures to control the execution of operating system (OS) privileged activities. The measures comprised providing central management of executing privileged commands and logging the activities on central server.

The second round of data collection was started after experimenting the control measures highlighted above. The teams in the previous round were approached to provide their feedbacks after implementing the updates and testing PaaS automation within the same organization.

The third round of data collection was triggered after summarizing experts' feedback and sharing the results of the second round with all participants. The proposed system in this paper was designed and built to address security and quality requirements. In this round, the design was shared with experts for their feedback. Based on their feedback, enhancements were added and final green light was received from experts to build the system for production operation.

B. Data

Critics were received during the first and second round. Table I provides PaaS automation critics summary. The proposed design in the next section was developed based on the critics and the requirements gathered during the first and second rounds of Delphi process. The design was drafted during the third round of Delphi process. Minor updates were added to the system design in this round. Final design was validated through experiment.

Table I.
Summary of Critics and Experiment Findings

#	Critic	Experiment Findings
1	Automation of application platforms is not reliable. Automation quality is low and cannot be used for mission critical systems.	Quality Assurance (QA) should be integrated in the process of PaaS automation development. The design includes lab environments to simulate scenarios and minimize the likelihood of failure. Control measures are added in the system to prevent direct update in production.
2	If automation is doing everything, labor will lose the technical skills and know how.	With new technologies, a shift in labor skills is required. The experiment findings related to this critic are summarized in Analysis and Discussion section.
3	The cost of building PaaS and the required automation is high.	Financial Return on Investment analysis is required to calculate cost against expected return. High level analysis is provided in Analysis and Discussion section.
4	Update of policies and procedures of managing and operating application platforms is required.	This is confirmed in the experiment. The update was mandatory to accommodate PaaS objectives.
5	Automating the deployment of applications involves multiple functional groups in Information Technology department.	The proposed design in this paper addresses Segregation of Duties (SoD) requirements. SoD is leveraged in the design to increase the level of automation quality and security.
6	Automation management server is empowered with privileged accounts and connected to applications and infrastructure components across the organization.	The design provides control for the privileged accounts. The power of PaaS production Orchestrator (PO) was contained by limiting the set of allowed operations. The system securely manages the distribution of automation artifacts. Unauthorized updates are detected by dual integrity control subsystem.
7	Specific experts expressed their ability to achieve the same results with traditional scripting approaches done at OS level.	Acceptable level of quality and security to implement mission critical PaaS is achieved in this experiment. To validate the claim, different approaches and designs should be experimented and compared objectively in future work.
8	Application platforms are updated regularly with new features, security and bug fixes. There is an overhead of keeping track of changes and updating automation.	A balance should be established between the value of update and automation development cost. Image based and plan based automation approaches were experimented. After the deployment of an older version, plans are executed to roll forward the platform to the required release.

IV. Proposed Control Design

In this section, quality and security management design for PaaS is provided. The design is divided into five layers based on roles and responsibilities to achieve the required quality and security control.

A. Layered Management System

The MS consists of Security Audit Layer, Infrastructure as a Service (IaaS) Management Layer, PaaS Management Layer, PaaS Automation Developer Layer and PaaS Consumer Layer. Segregation of Duties (SoD) concept is implemented to maximize security and quality control in mission critical systems without losing the opportunity of automating their deployment and routine operational activities. The five layers are reflected in Fig. 1.

Security Audit Layer includes Privileged Activities Management, Security Policies and Security Events systems. Security policies contain Delegated Privileged Commands (DPC)s which give delegate groups the authority to execute privileged commands. IaaS Privileged commands are executed in a controlled approach. First of all, IaaS administrator assess the impact of delegating and automating the command or infrastructure operation. Once assessed and approved, Security Analyst registers them and assigns them to a delegate group in a security policy within Security Audit Layer. Thereby, the infrastructure operation is utilized at PaaS layers as a self-service. Privileged Activities Management server enables a secure and central management of privileged commands. Security Events database logs the usage of DPCs remotely.

IaaS Management Layer is where infrastructure resources of the cloud are managed. Infrastructure resources include hypervisors, bare metal hosts, storage systems and network components. IaaS administrator manages this layer. His activities as a super user are controlled by Privileged Activities Management server. IaaS resources deployment and management are orchestrated by central IaaS orchestrator tool.

The third layer is PaaS Management Layer. This layer consists of PaaS Development Orchestrator (DO), PaaS Production Orchestrator (PO) and shared storage called Platform Images (PI). PaaS administrator manages these components to automate the deployment and management of PaaS applications. He also manages containerization platforms in contrast to hypervisors which are managed by IaaS administrator. PaaS DO consists of an application server and OS level Development Area. The application server is used to model PaaS automation processes by PaaS developer. It is also integrated with lab resources to conduct QA tests by PaaS developer in the next layer. Development Area is used to build OS level artifacts. The Orchestrator orchestrates PaaS automation in labs and provides pooling capability to logically isolate lab resources based on projects. Automation is developed and simulated in isolated environment before deploying it in production to verify security and quality. PaaS PO is generally similar to the DO. PO is connected to consumer assigned tenants and resources.

PI storage in PaaS Management Layer consists of Development Engine, Production Engine, Distributor, Repositories, Images and Shipment areas. PI storage is mounted in all PaaS resources. Development and Production Engines store automation artifacts. In order to release new automation artifacts from PaaS Development Area to Development Engine, Delegated Privileged Commands are used. PaaS Developer is delegated to update Development Engine using pre-approved delegated commands. These delegated commands verify integrity and analyze automation run time requirements. In addition, release event and content of developed artifacts are reflected in the Security Events system. The deployment to Production Engine is also protected. Only PaaS administrator is delegated to deploy through DPCs. This is enforced to verify the quality of automation before deploying them for production use.

The fourth Layer of the MS is PaaS Automation Developer which controls the development and QA of automation associated with PaaS. The control is enforced by system design. Developer cannot update a production artifact directly. Artifact represents automation logical unit. PaaS automation developers are PaaS automation experts who understand the requirement to deploy and manage applications' platforms.

The fifth layer is PaaS Consumer Layer. PaaS consumer can be SaaS service provider or Organization software developer [7]. PaaS consumer consumes PaaS automated services on his assigned resources through PO.

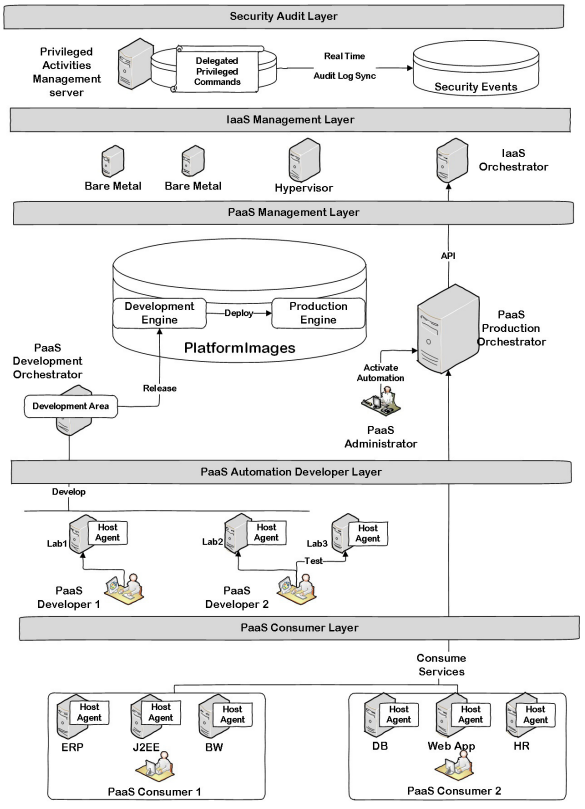

Fig 1. PaaS Management System

V. Experiment

The design was internally validated through experiment. It was implemented in a private CC data center.

A. PaaS Layers

PaaS orchestrators in PaaS Management Layer are implemented on Java platform with web user interface. Orchestrators are installed on LINUX OS. PI storage is based on Network Attached Storage (NAS). NAS is mounted with read only permissions on Developer and Consumer layers while it is mounted with write permissions on PaaS Management layer. This provides an additional layer of control at storage layer to prevent unauthorized update from a random cloud host. Lab in PaaS Automation Developer Layer consists of 14 servers and is divided into pools to simulate production environment and isolate developers' environments. A total of seven PaaS automation developers worked on the lab environment during the experiment. The developer models and defines automation processes to execute artifacts on the target managed cloud servers. Deployment of database, application server and web application server was simulated. Plans and artifacts were developed to patch and update middleware, database systems and application run time environment in PaaS.

PaaS Consumer Layer consists of 26 tenants hosted in 219 servers provided by IaaS layer. The hosts are integrated with PaaS PO through Host Agents (HA).

HAs are required by PaaS Automation Orchestrator in this MS. Secure HyperText Transfer Protocol (HTTPs) is used for the communication between PaaS Orchestrator and PaaS cloud resources. HAs are started automatically during the booting process of the servers and run with IaaS privileged accounts in the managed consumer servers. The agents digitally trust the orchestrator. Automation artifacts are whitelisted through definitions in the HAs. By that, HAs provide additional layer of control on cloud hosts.

B. IaaS Management and Security Audit Layers

In IaaS Management Layer, 90% of servers are virtual and managed by hypervisor. The remaining 10% includes bare metals. The bare metals host web applications and multi-tenant platform containers in the experiment. During the provisioning process of an IaaS server, HA is installed automatically as a sub-provisioning plan and PI NAS is mounted automatically. Artifacts definitions in Production Engine are reflected in the HA to achieve automated and direct integration with PaaS orchestrator.

In Security Audit Layer, security policies for DPCs were created. The policies are assigned to PaaS developers and PaaS administrators based on roles. DPCs include PI management commands to add new components or definitions, commands to deploy from Development Engine to Production Engine, and commands to rollout automation definitions from Engines to Cloud HAs. Contents of automation artifacts and definitions are written to Security Events database during the release, deploy or rollout of these automation units. Management of HAs is also achieved securely using DPCs by PaaS administrator. Events at HAs are monitored and logged in Security Events subsystem.

VI. Analysis and Discussion

In this section, critics collected as part of Delphi process are analyzed based on experimental observations. The experiment was conducted in CSP organization. Organizational factors may impose threats to the internal validity of the experiment such as management support and sponsorship.

Critic number one is related to the perceived service quality. PaaS automation cannot be guaranteed by 100% as observed in experiment. QA should minimize the probability of automation failure. Automation developer support is required to fix automation failure based on SLAs.

With respect to critic number two, the following was observed. PaaS automation requires deep understanding of how a platform is deployed and managed. PaaS automation developer needs to consider possible failure scenarios. These scenarios are simulated in lab. PaaS automation developer needs to be an expert in the platform being automated. After the deployment of PaaS automation, the developer is needed to troubleshoot, optimize and fix issues. Upon the release of a new platform version or patch, developer needs to review and update automation. In the experiment, modularity and reusability approaches were adopted. Based on observation, labor technical skills about application platforms were enriched. There was a shift in the type of work being done by the labor. Instead of doing a deployment task manually and sequentially, the work is shifted toward platform deployment automation analysis, PaaS automation development, PaaS automation QA, and PaaS automation lifecycle management. Throughput of one technical labor is increased.

Critic number 3 is related to the perceived relative advantage. Database platform deployment automation was analyzed from human hours' perspectives. To deploy 100 database systems manually, labor needs to works sequentially. Deployment of one database system manually requires 4 hours of work in average. In total, 400 human hours are required to deploy 100 servers. On the other hand, development and QA of automating database system deployment requires an average of 80 hours based on the study experiment. The created value is extended with the deployment of more servers while the initial investment cost from the CSP is the same. Breakeven is achieved after the deployment of the 20th database system. Additional benefit beside human hours saving include service agility. Deployment of the server automatically was done in less than 30 minutes. On the other hand, cost of operating PaaS MS should be considered. In addition, customer support service is required in case of automation failure. Accordingly, financial feasibility study is required to measure return on PaaS automation investment.

Critic number 4 is related to PaaS compatibility with the organization. Policies and procedures were updated to achieve the experiment in the organization. An example for that is handling the use of root user which is a privileged account at LINUX OS system level. By focusing on the com-

mon goal of providing competitive and secure services to customers and with organization management support, legacy policies were updated to fit with the proposed PaaS MS.

Critic number 5 can be related to organization size. The experiment was conducted in organization where OSs, database systems and application servers are supported by different functional units. Each team has a set of privileged accounts to manage its services. In the experiment, it was found that deployment of database requires OS and database privileged accounts. The proposed design described above resolved SoD issues associated with the use of privileged accounts owned by different functional group.

Critic number 6 involves the perceived security risk of PaaS. The unlimited power of PaaS Orchestrator was controlled using the proposed MS. Only authorized artifacts are allowed to be executed by the Orchestrator. Authorization is achieved through the definitions in the HAs. The distribution of definitions from PI to HAs is achieved securely by the MS. Integrity of the definitions and the associated artifacts is monitored through Dual Integrity check subsystem.

With respect to critic number seven, it is not believed that the proposed solution in this paper is the only possible management solution for PaaS. However, different solutions can be proposed and evaluated. Comparison between systems can be discussed in future research papers.

With respect to the last critic, it was confirmed in the experiment that there is an overhead of maintaining application platforms delivered by PaaS. In the experiment, flexibility, reusability, complexity and dependency automation properties [5] were incorporated in automation development to minimize the maintenance overhead. Modular and reusable components represent the logical units of PaaS automation plans. Parameter can be passed to these reusable units to control the use case. Also plan based automation and image based approaches were utilized. By that, a base image of the platform is maintained. Then, new functions, updates and patches are added to the platform using plan artifacts to roll-forward the platform to the desired version.

VII. Conclusion

Based on this study, managing mission critical systems in private PaaS Cloud is practical. The study revealed PaaS adoption factors which can be categorized into organizational, relative advantage, compatibility, security and quality factors. Organizational, compatibility and relative advantage factors are analyzed in the previous section. Security and quality technological factors are incorporated in the MS model and experimented. The proposed model is considered a novel MS for CSPs aiming to provide high level of security and quality standards. The study also gives PaaS cloud users an understanding of how systems can be managed internally by CSPs which should lead to better SLAs in the future.

Comparison with other models and technologies can be done in future research. It is also suggested to validate the model under public cloud and examine the forensic aspects.

References

[1] M. E. Alex and R. Kishore, “Forensics framework for cloud computing,” *Computers & Electrical Engineering*, vol. 60, 2017, pp. 193-205. doi:10.1016/j.compeleceng.2017.02.006

[2] S. Costache, D. Dib, N. Parlavantzas and C. Morin, “Resource management in cloud platform as a service systems: Analysis and opportunities,” *Journal of Systems and Software*, vol. 132, 2017, pp. 98-118. doi:10.1016/j.jss.2017.05.035

[3] S. N. Deshmukh and H. P. Khandagale, "A system for application deployment automation on cloud environment," *2017 Innovations in Power and Advanced Computing Technologies (i-PACT)*, Vellore, 2017, pp. 1-4. doi:10.1109/IPACT.2017.8245025

[4] M. Boschetti, V. Baglio, P. Ruiu and O. Terzo, "A Cloud Automation Platform for Flexibility in Applications and Resources Provisioning," *2015 Ninth International Conference on Complex, Intelligent, and Software Intensive Systems*, Blumenau, 2015, pp. 204-208. doi:10.1109/CISIS.2015.29

[5] J. Wettinger, V. Andrikopoulos, F. Leymann and S. Strauch, "Middleware-Oriented Deployment Automation for Cloud Applications," *IEEE Transactions on Cloud Computing*, vol. 6, no. 4, pp. 1054-1066, 1 Oct.-Dec. 2018. doi:10.1109/TCC.2016.2535325

[6] J. O. Benson, J. J. Prevost and P. Rad, "Survey of automated software deployment for computational and engineering research," *2016 Annual IEEE Systems Conference (SysCon)*, Orlando, FL, 2016, pp. 1-6. doi:10.1109/SYSCON.2016.7490666

[7] A. J. Ferrer, D. G. Pérez and R. S. González, “Multi-cloud Platform-as-a-service Model, Functionalities and Approaches,” *Procedia Computer Science*, vol. 97, 2016, pp 63-72. doi:10.1016/j.procs.2016.08.281

[8] J. Wettinger, V. Andrikopoulos, S. Strauch and F. Leymann, "Characterizing and Evaluating Different Deployment Approaches for Cloud Applications," *2014 IEEE International Conference on Cloud Engineering*, Boston, MA, 2014, pp. 205-214. doi:10.1109/IC2E.2014.32

[9] J. Wettinger, U. Breitenbücher, O. Kopp and F. Leymann, “Streamlining DevOps automation for Cloud applications using TOSCA as standardized metamodel,” *Future Generation Computer Systems*, vol. 56, 2016, pp 317-332. doi:10.1016/j.future.2015.07.017

[10] X. Lan, Y. Liu, X. Chen, Y. Huang, B. Lin and W. Guo, "A Model-Based Autonomous Engine for Application Runtime Environment Configuration and Deployment in PaaS Cloud," *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, Singapore, 2014, pp. 823-828. doi:10.1109/CloudCom.2014.80

[11] K. Kritikos, T. Kirkham, B. Kryza and P. Massonet, “Towards a Security-Enhanced PaaS Platform for Multi-Cloud Applications”, *Future Generation Computer Systems*, vol. 78, Part 1, 2018, pp 155-175. doi:10.1016/j.future.2016.10.008

[12] J. O. de Carvalho, F. Trinta, D. Vieira and O. A. C. Cortes, “Evolutionary solutions for resources management in multiple clouds: State-of-the-art and future directions,” *Future Generation Computer Systems*, vol. 88, 2018, pp. 284-296. doi:10.1016/j.future.2018.05.087

[13] Y. Jinzhou, H. Jin, Z. Kai and W. Zhijun, "Discussion on private cloud PaaS construction of large scale enterprise," *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, Chengdu, 2016, pp. 273-278. doi:10.1109/ICCCBDA.2016.7529570

[14] P. Mell and T. Grance, “The NIST Definition of Cloud Computing,” National Institute of Standards Technology, Gaithersburg, MD, USA, Technical Report, SP 800-145, 2011. doi:10.6028/NIST.SP.800-145

[15] M. Kostoska, I. Chorbev and M. Gusev, "Creating portable TOSCA archive for iKnow University Management System," *2014 Federated Conference on Computer Science and Information Systems*, Warsaw, 2014, pp. 761-768. doi:10.15439/2014F311

[16] L. Rodero-Merino, L. M. Vaquero, E. Caron, A. Muresan and F. Desprez, “Building Safe PaaS Clouds: A Survey on Security in Multitenant Software Platforms,” *Computers & Security*, vol. 31, Issue 1, 2012, pp. 96-108. doi:10.1016/j.cose.2011.10.006

[17] R. Dukaric and M. Juric, "BPMN extensions for automating cloud environments using a two-layer orchestration approach," *Journal of Visual Languages and Computing,* vol. 47, pp. 31-43. doi:10.1016/j.jvlc.2018.06.002