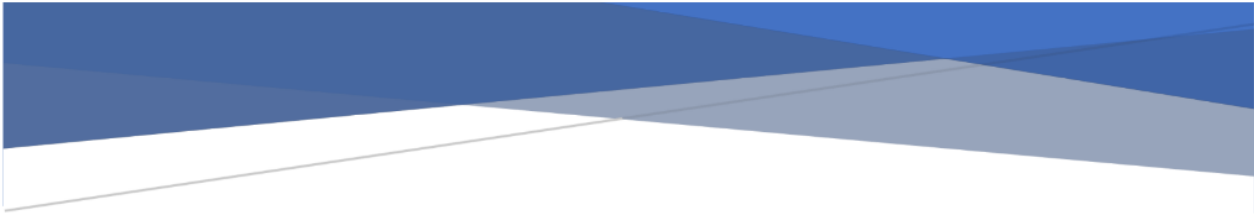


Emerging and Future Trends (Chris Morris)

by Christopher Morris

FILE	1572_CHRISTOPHER_MORRIS_EMERGING_AND_FUTURE_TRENDS__CHRIS_MORRIS__226879_358373606.DOCX (116.77K)		
TIME SUBMITTED	10-FEB-2020 06:20PM (UTC+0000)	WORD COUNT	2267
SUBMISSION ID	119880383	CHARACTER COUNT	12033



EMERGING AND FUTURE TRENDS

Chris Morris

Module code: ACCA7016
Lecturers: Michael Dacey and Tim Bashford

Contents

Abstract.....	1
Introduction	1
Findings	1
Platform as a Service.....	1
Container as a Service	2
Comparison	2
Docker	3
Benefits and drawbacks of Docker	3
Docker relation to PaaS, CaaS, and IaC	3
Infrastructure as Code (IaC)	3
Future usage of IaC	4
Kubernetes.....	4
Relation to Docker containers	4
Alternatives	4
Conclusion.....	5
References	5

Abstract

PaaS, CaaS, and IaC are growing trends in cloud computing, but other examples include Serverless Architecture and Edge Computing. Each trend has their advantages and drawbacks to consider, and which one is most ideal depends largely on an organisation's demands. In this case, PaaS and Serverless Architecture were chosen as the trends with the most potential.

Introduction

As cloud computing and cybersecurity continues to evolve, it results in trends that enhance its performance, including Platform as a Service (PaaS) and Container as a Service (CaaS). These are achieved via software such as Docker and Kubernetes. The objective of this report is to analyse

both the software and the trends, including how beneficial they are and any drawbacks.

This report will involve forming a systematic analysis of emerging trends within cloud computing. The strengths and weaknesses of each will help determine which has the most potential. Any other technologies not listed in the scenario but deemed worth mentioning will also be discussed.

Findings

Platform as a Service

PaaS is a variety of cloud computing that specialises in controlling computer platforms (executable software such as operating systems that effectively serve as the backbone of the system).

Only the application and data are handled by the user, with everything else (the operating system, storage, servers etc.) are handled by the manufacturer. [1] Having little variables controlled by the user means that it functions as an off-site mechanism.

An example of PaaS is Google App Engine, which combines clusters (a group of servers) into one platform. This is the period where PaaS first become noteworthy, at the time named Framework instead of Platform. It was created with the aim of combating users having to spend more time on the technicality of an application, which a platform will instead construct.

Normally, a PaaS contains three layers: programs, platforms, and infrastructure (any mechanics such as servers). There are two main ways they function, either privately (where data remains primarily

within the organisation) or publicly (where data is shared via being stored by third parties).

Private PaaS should be required when organisations require more control over their infrastructure (which makes them more secure due to more restrictions). In turn though, the servers must be configured manually, which can cost time and resources.



Public PaaS is the better choice if saving costs is the priority. Having fewer restrictions makes it more scalable and therefore more reliable than private PaaS. However, resources are shared, resulting in a higher risk of shortage.

If neither Private nor Public is deemed satisfactory, there is a compromise model called a Hybrid, which combines the elements of both Private and Public, therefore incorporating the strengths and flaws of both frameworks. [2]

The main advantages of PaaS overall are that, by directing so many features to be configured by management, accessibility is made quicker to users. Also, remaining a popular trend in cloud computing means that updates are common.

In contrast, the uneven distribution of features controlled by users compared to management means that further security reviews may be required to ensure enough control is maintained. Alongside this, the cloud features rely on the performance of the existing infrastructure to perform to its potential; they will not automatically increase speed, for example.



Container as a Service

CaaS allows users to control the application, data, and runtime of

containers (a structure that handles the code of software). [3] An example of CaaS is Kubernetes, which sandwiches its data between sections detailing data of management and security.

A CaaS model usually sees the infrastructure as the bottom layer, any virtualisation in the middle, and the containers placed at the top.

An example overall benefit of CaaS is its simplicity. Having containers that store the bulk of work can help save time, especially as tasks are spread out via load balancing to prevent stagnation.

This results in an increased cost, however, due to the amount of storage that containers require. Also, containers rely on network connections more than platforms, meaning that should a network disaster occur, containers are at a higher risk of data loss.

Comparison

PaaS and CaaS both allow users to manage application and data, but CaaS allow grants the ability to manage runtime.

The superior benefits of PaaS encompass its flexibility compared to CaaS. It provides a greater control of network traffic and contains fewer layers (which is considered an advantage due to more restrictions). CaaS, meanwhile, is more flexible in terms of compatibility and sharing data, as well as being able to complete more specific requests at a quicker rate due to said flexibility.

Of the two, it is determined that PaaS is the superior trend due to its more restrictive nature. However, a CaaS should still be highly recommended if working with programs such as Docker due to how

compatible container software is with each other.

Docker

Docker can be described as a simulation program capable of executing commands for containers. [4] It first came to prominence in 2013 in France, as an open-source container program. Docker specifically deals with handling computer objects. Its primary component is a daemon, the core feature that effectively serves as the central control area while running in the background.

The daemon stores containers and their previews, named images, in repositories. Repositories can be either private or public, and the main tool for creating images is "Docker Compose". Between the daemon and repositories serves a client layer where the user interface is based.

Benefits and drawbacks of Docker

Docker performs well in its scalability. Allowing users to define the scope makes it quicker to access without fixed parameters and simpler (if user is familiar with Docker) and cost-effective. Allowing manual input also means that Docker is compatible with other open-source container software such as Kubernetes. This makes sharing work across platforms easier and helps iron out flaws of applications.

In contrast, Docker's flaws include its lack of flexibility. It may not lend itself well to software that do not utilise containers. Said flexibility is also hampered by Docker needing to run on the same operating system as the device, result in no system-crossing. It also has a steep learning curve for those unfamiliar with open source

programming, and cannot function without the prior setup of platforms. [5]

Docker relation to PaaS, CaaS, and IaC

Docker has an apparent relationship with CaaS since it functions as container software, but still serves a role in PaaS, inspiring the hypervisors (group of virtual machines) seen in platforms. [2] In terms of IaC, Docker can assist with filling in any missing components, increasing the time that a container can be deployed for efficient usage.

Infrastructure as Code (IaC)

IaC can maintain control of data centres via configuration files rather than executable software. [6] An example of IaC is SaltStack, which specialises in manual configuration and adding of applications.

IaC was developed to settle demand for a cloud framework that builds upon the flaws of previous framework, including slow speed of gathering resources.

There are three main forms of IaC: Declarative, which specialises in determining the overall pathway; Imperative, which mainly examines the impact on the configuration files; and Intelligent, the overall logic of the configuration. Declarative is more useful in terms of testing (since it is a repeatable process), while Imperative is the more cost-effective choice. Intelligent, like Hybrid for PaaS, serves as a middle ground between the two. [7]

The reason why IaC has grown to become a popular cloud computing trend is because of its manual configuration. Users can set up an infrastructure themselves reducing cost and grants more control to them.

An opposite view is that IaC requires more knowledge of configuration in its setup, which can therefore result in more time being spent building its configuration. If a mistake is made, time will need to be spent tracking the source since the file is manually constructed.

Future usage of IaC

Going forward, IaC could potentially include features such as artificial intelligence (AI) in its design to assist with the structuring and containment of data. This will benefit people less trained with manual configuration as the risk of mistakes will be lower. As demand for cloud computing continues to grow, IaC will, in future, likely play a key role in providing a simple and quick service to handle said demand.

Kubernetes

Kubernetes is a container software that works automatically rather than manually. This results in all data remaining on-site rather than off-site like PaaS or CaaS. [8]

Kubernetes was first developed in 2003 to develop a cluster application with a large scope, but did not come to prominence until 2013 after being resurfaced by Google. An example usage of Kubernetes is to contain volumes of data about video games (e.g. time spent in a certain region).

Kubernetes' advantages comprise its compatibility, particularly so with other user-interface software such as Docker. Its high amount of storage is beneficial in being scalable. However, having more set parameters results in being more complex as control to the user is restricted. On top of that, the learning curve is steep for inexperienced users, meaning it takes more time to get used to.

Relation to Docker containers

Kubernetes can work together with Docker to accomplish the merging of virtual machines. If data is low on one end, more can normally be extracted from the other end to prevent a shortage. Docker and Kubernetes complement each other well to each other due to both being open-source programming containers; any changes can integrate easier as a result.

Alternatives

Alongside PaaS, CaaS, and IaC, examples of other emerging cloud computing trends include serverless architecture. [9] This involves the server being eliminated from the infrastructure and just the application being hosted by the third-party provider. FaaS (Function as a Service) is an example of a serverless architecture.

This trend is turning popular because of how much costs it saves. Eliminating the server means that less time is spent setting hardware, and can help pinpoint the full scale of a device's performance and scalability. However, any performance checks have to be more precise without a server to point out any flaws. Also, not having a server to fall back on in the case of a disaster can slow down the time it takes for an organisation to recover from one.

Another growing trend is Edge Computing. This entails connecting devices to the closest network edge (central area where the device registers an Internet connection) so that connecting to cloud computing is made easier through less time spent. [10] BMC Helix is an example software capable of providing Edge Computing.

Edge Computing is becoming favoured because of its productivity. It is designed

to connect users to cloud servers quickly, and helps bridge the gap between user and provider.

On the flip side, its negative features comprise of needing more hardware to get full use out of. While connection speed may improve, performance speed may not unless a device has been fine-tuned so that all benefits are maximised (e.g. storage is not overloading and can store data of images). Being placed on a more local scale also poses a security risk due to less restrictions.

Between the two alternatives, serverless architecture can be deduced as the more positive trend due to it increasing cost benefits and accuracy in finding mistakes, albeit manually.

Conclusion

Overall, understanding was made into the emerging trends of cloud computing. While PaaS and Serverless Architecture in particular were identified as the most promising future trends in the author's opinion, all three have their benefits and issues to consider, and which one is most useful depends on an organisation's criteria.

References

- [1] 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, pp. 63–72, 2016.
- [2] D. Rani and R. K. Ranjan, "A Comparative Study of SaaS, PaaS and IaaS in Cloud Computing," *International Journal of Advanced Research in Computer Science and Software Engineering*, vol. 4, no. 6, pp. 458–461, 2014.
- [3] M. K. Hussein, M. H. Mousa, and M. A. Alqarni, "A placement architecture for a container as a service (CaaS) in a cloud environment," *Journal of Cloud Computing*, vol. 8, no. 1, pp. 1–15, 2019.
- [4] C. Boettiger, "An introduction to Docker for reproducible research, with examples from the R environment," 2014.
- [5] T. Bui, "Analysis of Docker Security," 2015.
- [6] O. Lavriv, M. Klymash, G. Grynkevych, O. Tkachenko, and V. Vasilenko, "Method of cloud system disaster recovery based on 'Infrastructure as a code' concept," *14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering, TCSET 2018 - Proceedings*, vol. 2018-April, pp. 1139–1142, 2018.
- [7] M. Guerriero, M. Garriga, D. A. Tamburri, and F. Palomba, "Adoption, Support, and Challenges of Infrastructure-as-Code: Insights from Industry," *Proceedings - 2019 IEEE International Conference on Software Maintenance and Evolution, ICSME 2019*, pp. 580–589, 2019.
- [8] B. Burns, J. Beda, and K. Hightower, *Kubernetes*. 2018.
- [9] A. Pérez, G. Moltó, M. Caballer, and A. Calatrava, "Serverless computing for container-based architectures," *Future Generation Computer Systems*, vol. 83, pp. 50–59, 2018.
- [10] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge Computing: Vision and

Challenges," *IEEE Internet of Things Journal*, vol. 3, no. 5, pp. 637–646, 2016.

Emerging and Future Trends (Chris Morris)

GRADEMARK REPORT

FINAL GRADE

64/100

GENERAL COMMENTS

Instructor

PaaS (50) 28

IaC (25) 21

K8s (25) 15

Comments

=====

A fairly good description of PaaS and CaaS, but lacks some detail e.g. what exactly is a container? The comparison was a little naive, more should have been made of different scenarios where they could be applied and/or more could have been made of their benefits and drawbacks.

I liked some of the comparison of types of IaC. The discussion of future usage was very limited, but you did debate the causes and potential limitations might restrict its adoption..

The discussion of K8s was a little limited and there was no example of its usage.

More references required.



Comment 1

References



Comment 2

I think you need to justify this statement



Comment 3

references



Comment 4

References



Comment 5

Reference