Skip to Main Content

Emerging and Future Trends Report 1

by Greg Rees

FILE	16864_GREGORY_REES_EMERGING_AND_FUTURE_TRENDS_REPORT_1_226879_953955467.DOCX (97.69K)		
TIME SUBMITTED	03-FEB-2020 10:42PM (UTC+0000)	WORD COUNT	3634
SUBMISSION ID	119558613	CHARACTER COUNT	20349



Prifysgol Cymru
Y Drindod Dewi Sant
University of Wales
Trinity Saint David

Emerging and Future Trends

Report 1

Name: Greg Rees

Student number: 1905774

Programme: MSc Applied Computing

Module: Emerging and Future Trends

Date submitted: 3rd February 2020

Contents

1. Introduction 3
2. Platform as a Service (PaaS) and Container as a Service (CaaS) 3
2.1 Platform as a Service (PaaS) 3
2.2 Container as a Service (CaaS) 4
2.3 Advantages and disadvantages of Docker 6
3. Infrastructure as code (IaC) 7
4. Orchestration management using Kubernetes 9
5. Conclusion 11
References 12

List of Tables

Table 1: Advantages and Disadvantages of Docker 6

1. Introduction

Advancements in technology and the growing use of the cloud as a storage medium has resulted in a number of new ways that businesses can choose to develop and deploy their IT software solutions in the cloud.

Platform as a Service (PaaS), Container as a Service (CaaS) and Infrastructure as Code (IaC) are different methods of cloud computing that have soared in popularity over the last decade.

This report aims to explore the concepts of Platform as a Service (PaaS) and Container as a Service (CaaS) and identify any similarities or differences between them. Infrastructure as Code (IaC) will also be discussed and future usage of this technology evaluated.

Docker is the current market leader for container services and will also be explored in this report, as will Kubernetes which helps developers to manage and organise their Docker containers.

2. Platform as a Service (PaaS) and Container as a Service (CaaS)

2.1 Platform as a Service (PaaS)

Platform as a Service (PaaS) is a type of cloud computing that provides a suite of tools for software development over the internet. In traditional software development, developers and engineers are responsible for managing all resources such as infrastructure and hardware as well as managing software updates and security issues. PaaS strips this away these responsibilities by delegating them to PaaS service providers.

According to Krancher.O *et al* (2018), PaaS enables software developers to rapidly develop software applications, deploy and test them more efficiently and be able to scale them easily if required.

Being able to rapidly develop applications is achieved by removing the requirement for developers to manage the core infrastructure at the heart of the software, such as servers, storage and networking. The management of the application infrastructure is instead handled by the PaaS service provider, such as Googles App Engine or Amazon AWS. This then enables developers to concentrate solely on the development of their applications.

In addition to speed of development, Microsoft (2020) put forward that there are a number of other benefits from adopting a PaaS model for software development including:

- The removal of the need for suitably skilled staff as PaaS environments can offer new capabilities that existing teams can work with without having to possess the skills.
- The ability to develop applications for different platforms with greater ease and without having to have extensive knowledge of each platform.
- Being able to manage the whole web application life cycle of building, testing, deployment and management within the same environment.

Another benefit is derived from the flexible approach by some PaaS providers, meaning that cost savings can potentially be achieved as different tiers of resources can be requested and utilised as and when required. This removes the need for an organisation to have to maintain its own infrastructure such as servers and databases. If an application needs to be scaled, developers are able to utilise the resources of the PaaS providers depending on their requirements and are charged accordingly for what they use.

2.2 Container as a Service (CaaS)

Containers and virtual machines have become two of the most popular methods of deployment of software applications. Hussein *et al.* (2019) explain that there are differences between containerisation and virtualisation.

Virtualisation works by sharing the resources of a host machine and distributing them among the various virtual machines that are running on the host machine. A hypervisor, also known as a virtual machine monitor, is used to support the sharing of the resources such as memory. This method can prove to be inefficient as there is an additional level of software running above the host machine operating system potentially causing usability and stability issues as demand grows.

In an attempt to try and resolve the problems with virtualisation, advancements in technology have led to the development of containers for the purpose of distributing and executing software instead.

Docker has grown to become the market leader in container development. As explained by Docker (2020), containers are a unit of software that incorporates all code and related dependencies for an application to run efficiently and reliably in multiple computing environments. A container image is an executable package containing all resources needed to execute an application such as runtime, system libraries and code. During runtime of an application, containers become images.

One of the key advantages of Docker is that software applications that are run in containers will perform the same on all machines, irrespective of the core infrastructure. This is because containers segregate software from the environment they are deployed in.

The key difference between a virtual machine and a container is the absence of the hypervisor when using containers. In this method, containers share the same operating system kernel and resources as the host machine meaning that separate operating systems do not have to be launched for each instance of a virtual machine.

According to IBM (2019), Container as a Service (CaaS) enables developers to more efficiently upload, organise, run and manage containers for their applications.

As mentioned earlier in this report, PaaS is a development environment where all tools and resources are integrated and enable developers to create applications with ease. On the other hand, CaaS does not usually contain development tools and the focus is instead on managing and deploying containers. This is true of Docker, where applications are developed separately and then uploaded to a repository such as Docker Hub to be managed and distributed to users as necessary. In this regard, CaaS can be classed as different to PaaS.

IBM (2019) go on to explain that the approach to CaaS can be likened to PaaS in that resources can be added as needed such as scheduling capabilities. There is also a suggestion that CaaS is a subgroup of Infrastructure as a Service (IaaS) and can be placed somewhere between IaaS and PaaS.

Bigelow (2019) supports this view and explains that as the basic resource for CaaS is a container, instead of a virtual machine or similar bare metal host system which are commonly found in IaaS environments, it is more appropriate to place CaaS as a subcategory of IaaS.

IBM (2019) identify that there are a number of benefits of using CaaS including:

- Cost savings as developers only need to invest in the resources that are required e.g. scheduling capabilities. This is a similar approach to PaaS.
- Security is improved as containers are isolated from each other meaning that if an application in one container has performance issues it will not affect other containers.
- Containers can be easily scaled as services allow developers to run only the containers they need when they need them.
- Support from the container provider can result in secure and steady services.
- Container environments can be deployed rapidly as there is no dependency on an operating system allowing for new versions or updates to be released quickly.

2.3 Advantages and disadvantages of Docker

Containerisation has given developers a much more flexible, cloud-based alternative to using virtual machines. This has led to a number of container building platforms being developed and the current market leader is Docker.

Docker has certainly made a huge impact and gained serious momentum and sustained growth since it was first launched in 2013. Table 1 outlines the advantages and disadvantages of Docker as a development tool:

Table 1: Advantages and Disadvantages of Docker (ApacheBooster, 2018)

Advantages	Disadvantages
Despite being predominantly developed for use in a Linux environment, its popularity has meant it is becoming more established on other platforms such as Windows and Mac meaning it can appeal to more developers	The security aspect of containers can sometimes be overlooked during development and this could lead to hackers trying to explore this weakness and bring down applications
Rapid deployment of containers is possible and containers can be updated and reused as needed	Data stored within a container is lost when a container is terminated. There are some methods to backup and extract data from a container but these are not yet very smooth or user friendly
Containers tested in one platform environment will work in other environments as they are independent of the host operating system	Containers use resources more efficiently and can be faster however they still depend on the host machine and this can affect performance. If there are problems with the host system kernel this will affect the containers
Containers are lightweight, standalone and do not consume as many resources as virtual machines	Some features are still under development such as copying files from the host to the container
Docker has a large community following and a substantial bank of resources can be found to assist in creating containers	At the moment, Docker works best with applications run on the command line and is not yet fully suitable for application with graphical user interfaces

3. Infrastructure as code (IaC)

At the heart of all software applications is the infrastructure enabling it to run such as web servers, databases and networking configurations. Traditionally this has been managed by system administrators tasked with the responsibility for ensuring the applications and systems ran smoothly.

This manual approach was initially popular, however it led to problems such as a lack of consideration as to how environments would be managed, an over reliance on the system administrators preferred way of working and a subsequent lack of version control and audit trails showing changes to the environment causing problems if a system administrator left their position (Morris.K, 2016).

The advancement of cloud-based technology and methods of working has revolutionised the approach to managing infrastructure. Providers such as Azure Resource Manager and AWS CloudFormation now provide the tools and resources to assist developers with their software.

Artac *et al.* (2017) explain that in addition to the programming code being written to design a piece of software, code is also written for the infrastructure design as well. This can include things such as outlining the required dependencies and configuration requirements for software. This can be likened to the process involved producing a software application using a programming language such as Java for the program code and then including the additional Java libraries to produce a complete software application.

Artac *et al.* (2017) go on to explain that the purpose of using a IaC approach is to re-use successful configuration implementations in order to speed up software development. It can also be described as a blueprint that contains deployment specifications which can be replicated. The chosen provider is then responsible for controlling the infrastructure environment to suit the needs of the software.

According to Thorn Technologies (2018) and Plutora (2019), there are a number of benefits to an organisation adopting an IaC approach, including:

- **Speed and simplicity of development** – entire infrastructures can be enabled very quickly just by running scripts.
- **Consistency of configuration** – the creation of the infrastructure is standardised and can help to reduce human error. This should also result in less compatibility issues when running applications.
- **Mitigation of risks** – deployment of software updates can sometimes produce errors or issues that require fixing. Depending on the working methods of traditional system administrators, this may be difficult to fix. The ability of IaC to be version controlled allows all changes to the

configuration to be documented, logged and tracked so that any failed configurations can be identified and corrected.

- **Increased efficiency of development** – in the cloud environment the software development life cycle can be followed more efficiently. Developers are able to move through the cycle in multiple stages from development, testing to deployment. It is also possible to instruct IaC to control the running down of environments when they are no longer required, resulting in more productive teams and an organised cloud space.
- **Cost savings** – having the infrastructure managed by a provider allows development teams to be more productive and spend more time on the tasks that add value to a project. Cost savings can then be made on staff salaries.

IaC has certainly changed the landscape of software development and the benefits it offers has meant its popularity continues to grow. It is likely that more and more software development will move to an IaC approach in the future, however there are some considerations to take into account:

- **Changes in ways of working** - adopting IaC will result in changes to the way existing teams have managed software development. This could cause some anxiety amongst teams and must be managed appropriately. This could result in changes to an employee's job role, redeployment to another role if not longer required, or possible redundancy.
- **Training of staff** - there will need to be a significant financial and time investment in the training of staff to work in a new way. IaC needs to be fully integrated across the whole spectrum of software development in an organisation and ensuring all areas work together reliably and efficiently will be important.
- **Security of data** - if control of the infrastructure including databases is handed over to outside providers, the organisation may have some concerns about the security of its data on their systems. There will need to be robust security measures and a suitably comprehensive security policy to ensure the risks of data breaches can be mitigated.
- **Configuration drift** - this can occur when administrators make manual changes to configuration settings on a server but not to the IaC code itself (Morris.K, 2016).

4. Orchestration management using Kubernetes

Containers, such as those created by Docker, can be used to distribute and run applications very efficiently. However, containers are still prone to runtime errors and can stop working which can have knock on effects to users of the applications.

In some organisations there may be hundreds or even thousands of containers running which need to be managed effectively. As the use of containers increases and multiple container instances are being run there needs to be a coordinated way of managing and organising them, also known as "orchestrating".

Docker has its own native tool to manage this process called Docker Swarm. For beginners and smaller deployments, Docker Swarm may be the best choice. However, IBM (2018) put forward the point that Docker Swarm is not as functionality rich as some of its competitors, including Kubernetes.

The Kubernetes Authors (2019) explain that Kubernetes originated from Google and first launched in 2014. It provides developers with a framework that will allow multiple containers to run robustly as it is able to manage aspects concerning the automating, deploying, scaling and monitoring of containers. Microsoft (2019) go on to add that Kubernetes brings other benefits such as performing load balancing, managing resource application, health checking individual resources and being able to self-heal apps by automatically restarting containers. Microsoft (2019) cites the three primary reasons for using Kubernetes as portability, scalability and extensibility.

When deploying a container, developers are able to specify values for the environment such as CPU power, RAM, storage and Kubernetes will take care of all of these requirements automatically. If a container should fail for any reason, Kubernetes will attempt the self-heal process which is likely to result in starting a new container to replace the one that had failed.

The benefits of Kubernetes help to remove the burden of managing containers from developers who are then able to spend their time being more productive elsewhere in the software development life cycle.

IBM (2019) explain that the architecture of Kubernetes includes:

- **Clusters and nodes** – each cluster contains worker nodes which work together to deploy, run and manage applications, and one single master node that manages the worker nodes. Developers would have specified their requirements when setting up the container and the master node is responsible for responding to these and distributing resources appropriately.

- **Pods** – pods are groups of containers that have access to the same resources in a network. If there is an increase in traffic to a particular container , Kubernetes deploys pods to other worker nodes in the cluster. If a worker node goes down for any reason, Kubernetes can quickly start new pods on functioning worker nodes to ensure continuity of service.

There are many scenarios where Kubernetes has been used effectively to change the management of software development. One such example is from The New York Times and is explained by The Kubernetes Authors (2018).

Initially, The New York Times had invested in its own private data centres and took responsibility for managing their own infrastructure. As technology moved ahead, the company looked into using the Amazon cloud and moved some non-essential applications to the platform, whilst continuing to operate their own data centres for essential applications. This approach brought benefits, however they realised they were simply using the Amazon platform as another data centre. In addition, they had a similar approach to their virtual machines as Kubernetes does to containers which they were managing themselves in-house. Eventually, an investigation into becoming more cloud native identified that Kubernetes could be a suitable way forward and allow them to move away from the burden of maintaining their own virtual machines.

In 2017, the first Kubernetes based application was launched by The New York Times and originally it served only 1% of the total website traffic. The following two years saw a gradual and phased changeover resulting in now having almost 100% of the website applications running on Kubernetes.

An immediate effect from doing this was experienced in the speed of delivery of applications. Deploying Docker images and containers instead of activating virtual machines saw a reduction in deployment time from up to 45 minutes to down to a couple of minutes with Kubernetes.

It resulted in a move away from infrastructure being managed by the software engineering team to more autonomy within individual teams. This has developed a culture of sharing best practice and has sped up the software development process. The New York Times is now looking to explore what other benefits they can achieve from cloud technologies in their business model (The Kubernetes Authors, 2018).

5. Conclusion

The cloud technologies such as Platform as a Service, Container as a Service and Infrastructure as Code have certainly reshaped the way that software applications are developed and managed.

Platform as a Service has allowed organisations to move away from having to manage and maintain their own development environments, whilst Infrastructure as Code has enabled organisations to shift from providing and supporting their own infrastructure to delegating the responsibility out to IaC providers. Numerous benefits of both of these approaches have been identified, specifically the saving of costs and the improvements in efficiency that can be achieved.

The rapid growth of Docker since its launch in 2013 has shown that more and more development teams are recognising the benefits that developing software using container technology can bring. The added ability of using Kubernetes to manage and track multiple Docker containers simultaneously has made this technology more appealing.

There are now many different approaches that organisations can make to the development of their software. The pace of change is unlikely to slow down in the future and this could cause development teams some problems when choosing an approach to adopt. There are many issues to consider and any changes will have an effect on multiple parts of the organisation.

Changing from one way of working to another is far from a simple process and will require significant investments in time and budgets as well as ensuring that developers are sufficiently trained and competent in the use of the tools and resources available. Organisations will need to ensure that all factors are taken into account and feasibility studies conducted before committing to a new software development approach.

References

ApacheBooster, 2018. *What are the advantages and disadvantages of Docker?.* [Online]
Available at: https://apachebooster.com/blog/what-are-the-advantages-and-disadvantages-of-docker/
[Accessed 8 January 2020].

Artac.M, et al., 2017. *DevOps: Introducing Infrastructure-as-Code.* Buenos Aires, IEEE/ACM 39th IEEE International Conference on Software Engineering Companion, pp. 497-498.

Bigelow.S, 2019. *Containers as a Service (CaaS).* [Online]
Available at: https://searchitoperations.techtarget.com/definition/Containers-as-a-Service-CaaS
[Accessed 8 January 2020].

Docker Inc., 2020. *What is a Container? A standardized unit of software.* [Online]
Available at: https://www.docker.com/resources/what-container
[Accessed 8 January 2020].

Hussein.MK, Mousa.MH & Alqarni.MA, 2019. A placement architecture for a container as a service (CaaS) in a cloud environment. *Journal of Cloud Computing,* 8(7), pp. 1-15.

IBM Services, 2019. *What is Containers as a service (CaaS)?.* [Online]
Available at: https://www.ibm.com/services/cloud/containers-as-a-service
[Accessed 7 January 2020].

IBM, 2018. *Docker Swarm vs. Kubernetes: A Comparison.* [Online]
Available at: https://www.ibm.com/cloud/blog/new-builders/docker-swarm-vs-kubernetes-a-comparison
[Accessed 8 January 2020].

Krancher.O, Luther.P & Jost.M, 2018. Key Affordances of Platform-as-a-Service: Self-Organization and Continuous. *Journal of Management Information Systems,* 35(3), pp. 776-812.

Microsoft Azure, 2019. *What is Kubernetes?.* [Online]
Available at: https://azure.microsoft.com/en-gb/topic/what-is-kubernetes/
[Accessed 8 January 2020].

Microsoft, 2020. *What is PaaS? Platform as a service.* [Online]
Available at: https://azure.microsoft.com/en-gb/overview/what-is-paas/
[Accessed 8 January 2020].

Morris.K, 2016. *Infrastructure as Code: Managing Servers in the Cloud.* 1st ed. California: O'Reilly.

Plutora, 2019. *Infrastructure as Code: What Is It, and Why Should My Engineers Care?.* [Online]
Available at: https://www.plutora.com/blog/infrastructure-as-code
[Accessed 9 January 2020].

The Kubernetes Authors, 2018. *CASE STUDY: The New York Times: From Print to the Web to Cloud Native.* [Online]
Available at: https://kubernetes.io/case-studies/newyorktimes/
[Accessed 9 January 2020].

The Kubernetes Authors, 2019. *What is Kubernetes.* [Online]
Available at: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/
[Accessed 8 January 2020].

Thorn Technologies, 2018. *Infrastructure as Code: 5 Reasons Why You Should Implement IaC Now.* [Online]
Available at: https://www.thorntech.com/2018/01/infrastructureascodebenefits/
[Accessed 8 January 2020].

Emerging and Future Trends Report 1

GRADEMARK REPORT

FINAL GRADE

77 /100

GENERAL COMMENTS

Instructor

PaaS (50) 33

IaC (25) 22

K8s (25) 22

Comments

=========

Quite a nice discussion of PaaS, CaaS and IaC with some good comparison points made. A few more references required.

Good discussion of K8s and the example provided was very good.
We prefer numbered referencing.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

Comment 1

Reference

PAGE 5

PAGE 6

Comment 2

References

PAGE 7

PAGE 8

Comment 3

references

Comment 4

References

PAGE 9

PAGE 10

PAGE 11

PAGE 12