

Cloud Computing Models, Containerisation Technologies and Orchestration

by Paul Lawrence

FILE	3346_PAUL_LAWRENCE_CLOUD_COMPUTING_MODELS__CONTAINERISATION_TECHNOLOGIES_AND_ORCHESTRATION_226879_260184498.DOCX (933.99K)		
TIME SUBMITTED	03-FEB-2020 11:58PM (UTC+0000)	WORD COUNT	6671
SUBMISSION ID	119562237	CHARACTER COUNT	42470

Cloud Computing Models, Containerisation Technologies and Orchestration

Paul M Lawrence BSc (Hons) – MSc Postgraduate Student - Computer Networks and Cybersecurity
University of Wales Trinity St David, Swansea Bay Campus SA1
plawrence.152216@student.uwtsd.ac.uk

Abstract – This research paper critically evaluates cloud service technologies and Cloud Service Models (CSM); Platform as a Service (PaaS), Container as a Service (CaaS), Infrastructure as Code (IaC) and orchestration technology Kubernetes. A Comparison between PaaS and CaaS is performed and analysed for their relevance within today's modern cloud services market. The industry leading containerisation technology Docker is also critically evaluated for its advantages and disadvantages within a business's modern cloud architecture.

Keywords-component; *PaaS, CaaS, IaC, Kubernetes, Docker, Containerisation, VM, Hypervisor, Virtualisation, CSM, CSP*

Introduction to - Cloud Computing Models, Containerisation Technologies and Orchestration

Within modern day computing, aggregation of computer hardware resources and their underlying network infrastructure is fast becoming the norm. The centralisation of software applications and network services within datacentres that are delivered to end clients via the Wide Area Network (WAN) [1] i.e. the Internet, within a modern network architecture is called; 'Cloud Computing'. [2]

Cloud computing allows for large amounts of a computer's hardware resources to be aggregated within datacentres located in various locations around the globe, that are predominantly owned and maintained by Cloud Service Providers (CSP) [3]. The industry leaders within the CSP market are; Amazon Web Services (AWS) [4], Microsoft Azure [5] and Google Cloud [6].

A small portion of the vast computing resources housed within a CSP datacentre(s) are leased by businesses and/or generic end users for a nominal fee, in the form of a Virtual Private Cloud (VPC) [7], that provides security segmentation between various organisations utilising the CSP datacentre infrastructure. Each VPC is billed based on the amount of hardware resources that are utilised by end clients/customers of the company such as; Central Processing Unit (CPU) cycles [8], Hard Disk Drive (HDD) [9] storage space, volatile Random Access Memory (RAM) [10] Gigabytes etc which are calculated by the CSP and then invoiced within monthly cycles to the client. Cloud computing negates the need to purchase and maintain in-house servers which is economically expensive for companies.[11]

This research paper aims to deliver an insight within the evolving technologies used within a cloud architecture along with the umbrella of cloud business models these technologies reside within, that are referred to as 'X as a Service' (XaaS).[12]

The focus of this paper will be within Platform as a Service (PaaS) [13] and Container as a Service (CaaS) [14] along with the industry leading open-source Linux CLI containerisation framework; 'Docker' [15] and the emerging Cloud Service Model (CSM) Infrastructure as Code (IaC) [16]. Open-source orchestration software 'Kubernetes' [17] will also be researched and discussed within the impending sections of this paper.

I. PLATFORM AS A SERVICE

Traditionally PaaS refers to a centralised cloud software development framework within DevOps, that comprises of CSP vendor specific tools that restricts developers to only a subset of CSP development tools, that offer a limited number of programming languages and a single OS for developers to utilise within the development of their software platform, in order to create and deploy a customised website application within the cloud via an Application Programmable Interface (API) [18] located within a CSP datacentre known as the management node. The CSP API is accessible by means of a website browser, during the implementation, testing, deployment, data analytical and continuous development stages of a cloud software applications lifecycle [19].

Within a traditional PaaS CSM, the developer retains management and ownership of the software application, along with access to the end user and meta data generated by the platform, but crucially the developer does not have access to the underlying CSP datacentre compute and network abstraction layers in which the platform is housed, such as; runtime libraries, middleware, OS, virtualisation process, storage configurations and core network configurations. [20][21]

The aforementioned abstraction layers are all managed by the CSP, which is a drawback for developers whom need granular control of the underlying functionality of the software, such as;

- OS specific architecture.
- DevOps programming languages specific to a developer's skill set.
- Virtualisation technology utilised.
- Non CSP vendor open-source and/or customised DevOps tools.

- Specialised runtime libraries.
- Specialised middleware.

Traditionally PaaS utilises a type-1 hypervisor [22] within the virtualisation layer that is managed by the CSP that comprises of Virtual Machines (VM) [23], which is very resource intensive due to a full OS being utilised within each VM to perform only a limited number of tasks for the software to function. The majority of PaaS software applications do not require the myriads of libraries, programs and tools that are packaged within an OS by default, that run a vast amount of background tasks in order for the host OS within the VM to function correctly. [24]

VM will be discussed in further detail within the *Container as a Service* section of this paper.

The lower level abstraction layers within the compute and network stack of the CSP, are reserved for solutions architects whom have access control as far as the OS abstraction layer within the underlying network infrastructure housed within the CSP datacentre, along with network configuration settings such as; load balancing, Internet Protocol (IP) address management,[25] Dynamic Name System (DNS) [26] and security configurations. This is considered the highest level of control within a cloud architecture which is a separate CSM to PaaS that is called; Infrastructure as a Service (IaaS) [27].

The end user of a PaaS software application will perceive it as Software as a Service (SaaS) [28] because they will not have any access to the application source code or user and meta data generated by the software, for data analytics and/or application change controls [29] within the CSP API, this is reserved for DevOps [16] PaaS software engineers whom maintain the application as previously discussed. Therefore, PaaS can be defined as a software specific cloud development platform. [30]

A comparison of the aforementioned cloud service models and the levels of control within each of the compute and network abstraction layers can be found in *figure 1*.

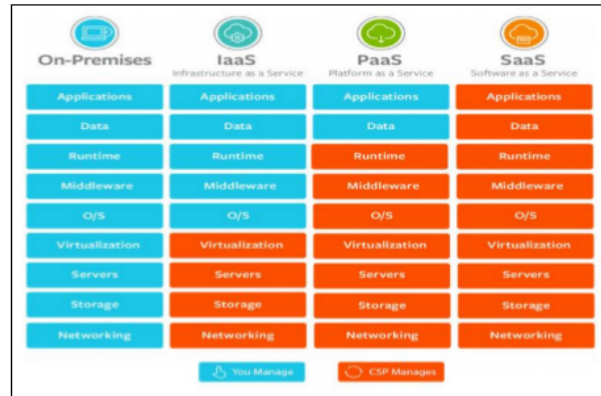


Figure 1: Cloud Service Models Abstraction Layer Management Diagram [31]

As *figure 1* illustrates, PaaS permits developers to manage their software applications and aggregate the data that is generated, collected analysed and correlated by the platform for real-time data analytics [32] via the CSP vendors API remotely for continuous change controls of their software platform as previously discussed.

II. CONTAINER AS A SERVICE

CaaS can be considered as a subset of IaaS rather than PaaS because even though CaaS can provide a cloud software platform for many end users and developers to utilise around the globe similarly to PaaS. However, CaaS can not only be utilised for development and deployment of software applications but it can also be utilised for networking services similarly to IaaS but without the full environmental overhead and high hardware utilisation of a VM to provide distributed 'Micro-Services' from a centralised location. Therefore, CaaS is not restricted to a software only cloud solutions. [33] [34]

Within CaaS the full stack DevOps engineers have granular control of the lower abstraction layers, down to the OS level network and security functionality equally to IaaS, with the caveat being that CaaS is technology specific to containerisation technology at the virtualisation layer that is not bound to CSP vendor specific tools.[35]

The main reason container virtualisation technology has gained a lot of traction within the IT industry in the last 3-5 years, is because of the vast hardware resource efficiency the technology provides in comparison to a type-1 hypervisor virtualisation architecture. [36]

For example, if an organisation such as a university wanted to provide software/network service to each specific department within a Campus Area Network (CAN), multiple VM would need to be utilised for each of the desired software/network services, which would use a very high amount of hardware resources, incurring large CSP fees for hardware resources that are not being utilised which is not economically efficient.

Therefore, containerisation is a lot more hardware efficient in the fact that as opposed to VM, containers are slim customised

versions of the most efficient open-source OS Linux variants, that only contain the required software dependencies that are essential for each specific service to function, that are compiled into a containerisation 'Image', thus only utilising the minimum hardware resources needed to achieve software and/or network service functionality that can be deployed in seconds, that is why containerisation is also referred to as a 'Micro-Services' solution. [37]

Furthermore, containerisation technology also provides universal compatibility, portability, optimisation and functionality of the software and/or network services for distribution to remote developers working within a DevOps project team regardless of the software dependencies installed within the host OS. This portability feature of containerisation also prevents businesses from being bound to one CSP after a platform has been created. [38]

A visual comparison of a virtual machine and container layers of abstraction with the compute architecture can be found in figure 2.

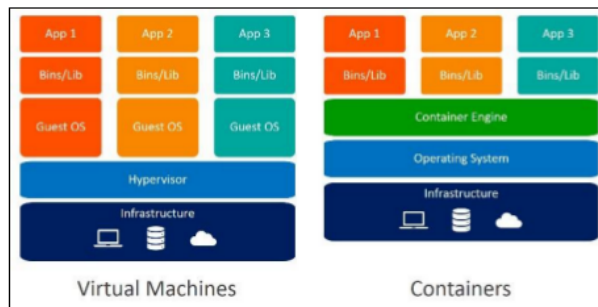


Figure 2: Virtual Machine vs Container Abstraction Layer Comparison [39]

As figure 2 depicts, not only is each container being utilised by the parent OS which reduces hardware resource overhead significantly, but it also maintains segmentation from other containers housed within the host OS at within the VPC, similarly to a VM. This segregated architecture is achieved by utilising a lightweight container management engine that is most commonly referred to as a 'Daemon' [40], that provides high efficiency and equally important security to each independent application and/or network service being utilised.

The industry leader within containerisation framework technology is the open-source Linux based containerisation framework; 'Docker' [41] that comprises of many features for customising various OS, software applications and/or network services using the Docker frontend CLI client.

A key feature within Docker is the rapid semi-automated process of creating a container image utilising a 'Docker' file, which is a configuration script native to Docker. A YAML [42] script can also be configured to commission the creation of multiple containers from a single and/or multiple image using a built in Docker tool, called 'Docker-Compose' [43]

Containers can also be networked within 'Docker' by creating a virtual network within the framework that allows granular configuration of; private IP address management, port exposure, Domain Name System (DNS) 'A' records for each container within the isolated Docker network so that the 'Micro-Services' [44] can communicate. This process can also be semi-automated using 'Docker-Compose' [45]. [46]

An example of 'Docker-Compose' industry use case can be found in Appendix-A of this paper.

The key to Docker's efficiency is the utilisation of Linux OS [47], libraries, services and a myriad of open-source tools as previously discussed, that are available to DevOps engineers to create and compile customised compiled container images of network/software solutions that are 'Containerised' for rapid deployment of software applications and/or network services to end users either locally or remotely from a CSP datacentre. Multiple containers can be made from a single Docker image that is utilised for scalability of a service. [48]

Another key feature within Docker is its repository called 'Docker-Hub' [49] that comprises of precompiled lightweight images that contain over 100,000 open-source tools and 'Micro-Services' such as; preconfigured SQL databases, stripped down versions of Linux OS etc. That can be downloaded via the Docker client CLI from the repository for developers to instantly utilise within their project and/or customise the image for their project requirements. [50][49]

Precompiled and configured container base images provide an instant working environment for developers negating the need of configuring tedious environmental variables for OS or open-source applications. For example, when a developer implements an SQL database into a webserver environment whom does not have the time and/or knowledge to configure the database he/she can just 'Pull' the preconfigured SQL database from Docker Hub via the CLI in seconds. [50][48]

A CaaS cloud solution not only provides accessibility to local developers whom may choose to work remotely away from a company's office for personal and/or business logistical reasons but CaaS unlike PaaS allows a business and/or startup to attract talented IT professionals from around the globe whom possess various skill sets within; software engineering that have specific programming language preferences, Linux OS systems engineering, database management and website development without them having to relocate to a different country or state where the company's head office is located.

However, one of the disadvantages of Docker is that the native orchestration tool; 'Docker-Swarm' [51] has very limited functionality within its fault tolerance feature, very limited container health monitoring and data analytics that does not include advanced learning or include an auto-scaling feature. [52] [53] [54][55][56]

Furthermore, persistent data storage volumes within Docker Swarm can be very complicated, tedious and clunky to configure and centralise within the host OS because by default, when a container is stopped, all data generated within the container is permanently lost. There is no object-store backend for container data centralisation or backup functionality, so configuring data persistent mount volumes is the only viable option within Docker Swarm that is very complicated and unreliable. That is why when compared to Virtual Hard Disks (VHD) [57] within VM Docker need to develop a more robust centralised data storage solution.[58][59][60]

Third party orchestration framework is recommended to mitigate these aforesaid issues such as; Kubemetes [17] that will be discussed in further detail within the *Kubernetes* section of this paper.

Another disadvantage is utilising Graphical User Interface (GUI) [61] applications natively within x1ldocker, that has many bugs and heavily impedes functionality of software due to the Linux CLI architecture that 'Micro-Services' within Docker was designed to be used with. [62][63]

A potential data security and availability flaw within Docker exists within its architecture, because unlike VM each of the containers within the host OS utilise the same kernel that could become corrupt and cause a Denial of Service (DoS) to the end clients. Also a malicious shell code thread injection at the kernel could also force a DoS, although this is highly unlikely as VPC services within a CSP datacentre are load balanced between multiple OS and servers to provide fault tolerance. [64][65]

The of these issues could be mitigated within future releases of Docker as it is a constantly evolving framework with a large open-source community of contributors that is becoming very popular within the IT industry.

Moving forward within cloud computing, it can be argued there is a use case for PaaS that is suited to software developers whom create very large software applications that cannot be distributed between 'Micro-Services' containers within CaaS and/or do not have multi-disciplinary skills within systems network/engineering that would prefer the underlying abstraction layers to be managed by the CSP. Furthermore, as previously discussed graphical software functionality with x11Docker service does not function very well within a current CaaS CSM as it does within PaaS VM, so this flaw within docker could deter software developers utilising the current version Docker.

III. INFRASTRUCTURE AS CODE

IaC has become a very popular concept within cloud computing as more companies utilise the cloud for their core network/software platform infrastructure necessities, whom have medium-to-large network environments and specific business requirements. These cloud virtual ecosystems need to

be created, deployed maintained, managed and scaled effortlessly and efficiently. [16]

A lot of IaC tools have matured over the past 3 years that facilitate the automation and management of a businesses data controls and data assets within a SME and large corporations cloud infrastructures around the globe.

The most efficient IaC implementation is the declarative software engineering paradigm [66] approach, as opposed to the imperative software model, to fully achieve an automated provisioning of a heterogenous cloud network infrastructures [67] configuration and management of the underlying virtual network environment, such as; servers, router, switches, firewalls, OS, containers, VM and the underlying CSP hardware infrastructure that the virtual infrastructure is built upon. [68] [67]

IaC allows for rapid development, deployment and scalability of cloud-based applications and/or network service platforms that can be applied to any of the aforementioned CSM discussed within this paper. The emerging IaC CSM and its technologies is a highly efficient approach to cloud DevOps in particular, that minimises human error and significantly decreases the cost of cloud based environments of network/application services by; automating the process, maximising hardware resource efficiency and achieving configuration compatibility version parity across the entire infrastructure/platform utilising 'Full Stack' engineers as opposed to specialist IT professionals, to attain ease of management, high availability, fault tolerance and scalable cloud platform solutions. [69], [70]

An IaC cloud software and/or network platform is created as an immutable network environment, to maintain the fulcrum of network stability across software versions and tools utilised within the provisioning of a cloud infrastructure. This approach guarantees cloud platform stability across the entire network that it is serving when implemented correctly.[71]

Furthermore, an immutable approach is most commonly utilised as opposed to a mutable architecture, because of the ease in which a network environment can be created and deployed within the cloud, using IaC technologies which would be transparent to the end users of the application and/or network services the platform is providing, in the event the environment needs to be modified and rebuilt for change control procedures of a platform within the Continuous Integration / Continuous Delivery (CI/CD) [72] pipeline.[71]

The most efficient IaC programming approach is the declarative, structured data modeling architectural method, that describes the desired granular final state parameters of a cloud environment that is being developed, by defining the logic of the structure and elements within the design without imparting the control flow in which the components within the system are built. As opposed to the imperative IaC software engineering

architectural method, that is less efficient compared to a declarative approach. [66], [73], [74]

Therefore, a procedural approach to IaC can be viewed as a semi-automated process akin to the conventional Born-Again Shell (BASH) [75] and Python [76] network/systems engineering configuration scripting because of the step-by-step method this system utilises.

Furthermore, the IaC declarative method utilises data modeling programming languages such as; YANG, XML, YAML, JSON etc [77] that are implemented utilising specialist IaC tools to achieve full construct automation of a cloud platforms environmental variables. These environmental variables are then parsed in the form of machine-readable definition files that contain configuration and deployment program scripts to the CSP Representational State Transfer (REST) [78] API in the form of a RESTful [79] transaction.

The configuration, deployment and management data scripts are created by utilising high level programming languages such as; Python, Java, PHP, C#, Pearl Go etc. [80] to achieve automation within a cloud platform as opposed to manual CLI syntax configurations within esoteric vendor specific languages such as; Cisco iOS, Microsoft PowerShell, Linux BASH etc.

IaC DevOps open-source, proprietary tools and technologies can be broken down into two distinct categories as follows;[81] [82]

Configuration Provisioning (CP) Tools; This category of IaC DevOps tools are utilised within the initial creation of the environmental variable constructs, that describe how the platform should function along with configuration and deployment of the platform. The industry leading tools and their data modeling and high-level programming languages that are utilised within the IaC platform are: AWS CloudFormation (JSON and YAML) [83], Terraform – ((Hashicorp Configuration Language (HCL) which is compatible with JSON)) [84]and Ansible – (Python) [85].

Configuration Management (CM) Tools; This category of DevOps tools is utilised to continuously manage and autonomously maintain software/network services within the cloud that includes; change controls for configuration updates/patches, scaling, monitoring of node health to maintain equilibrium within a system, along with logging. The industry leading tools within this category are; CFEngine [86], Chef [87], Puppet [88], Saltstack [84].

The abovementioned tools and their categories are collectively referred to as Continuous Configuration and Automation (CCA). A Data Flow Diagram (DFD) showing the lifecycle of a generic IoC can be found in figure 3.

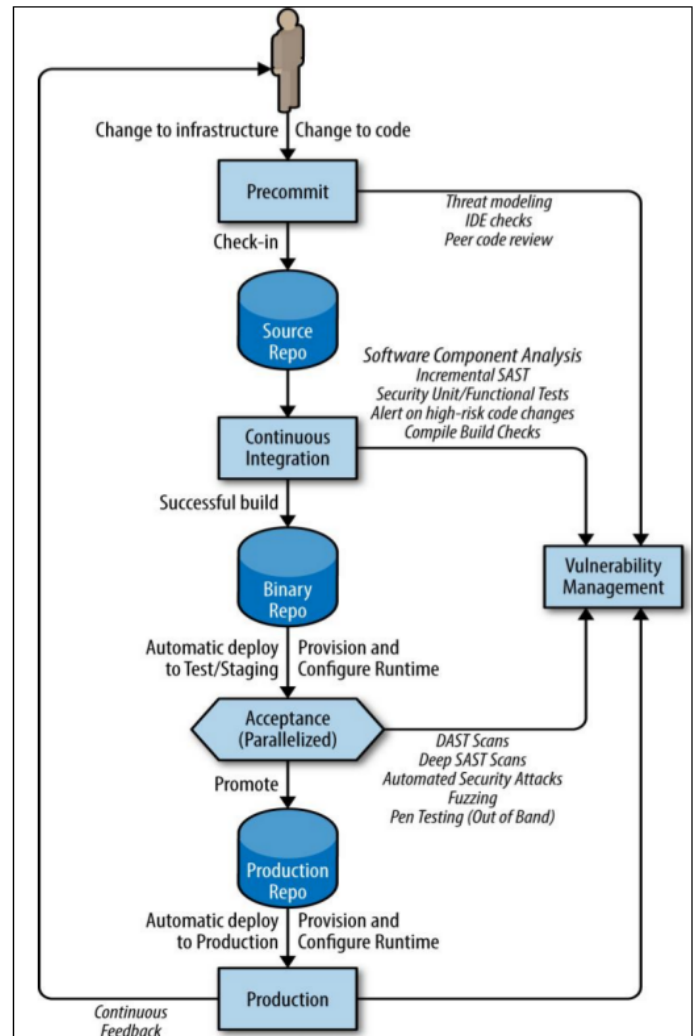


Figure 3: Generic IaC DFD Continuous Integration Lifecycle [69]

Docker and Kubernetes can be considered as IaC technologies within CaaS due to the data modeling YML and YAML data modeling programming languages that are utilised to automate the creation, deployment and management of an orchestrated 'Micro-Services' cloud environment. Kubernetes will be discussed in further detail within the following section of this paper.

IV. KUBERNETES

Kubernetes is an open-source orchestration framework created by Google that is utilised to create an automated virtual cluster, that manages a large number of virtual worker nodes within a workflow. Kubernetes is utilised to automate the management of the CI/CD pipeline within virtually distributed systems, which is very resilient via its CLI API located at the control plane, such as; 'Docker' containerised applications. [89]

A visual representation of how Kubernetes interacts with Docker within the virtualisation abstraction layer can be found in figure 4.

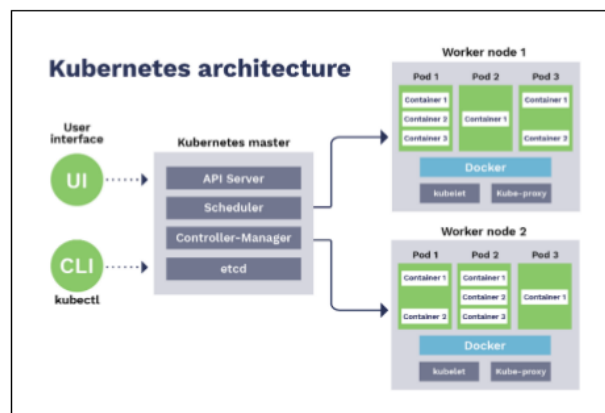


Figure 4: Docker and Kubernetes Integration Architecture

Kubernetes provides a declarative [90] approach to ‘Micro-Services’ automation. A group of containers that are deployed on the same host machine that share system resources are referred to as a ‘Pod’ that is managed by a worker node. All worker nodes are centrally managed by the Kubernetes master node.[91]

Each node within a pod can be configured to automate the distribution of hardware resources such as CPU and RAM utilisation, to maximise container performance efficiently at the hardware abstraction layer, similarly to how a hypervisor allocates resources to VM. [92]

Every deployment and management functions can be automated within Kubernetes by utilising YAML IaC configuration scripts at the Kubernetes master server API. [91]

Due to the lightweight architecture of a container and rapid start-up times, Kubernetes has fast become a vital framework with the advent of CaaS, that can be utilised for businesses whom are transitioning to a fully integrated ‘Micro-Service’ virtual ecosystem that can have hundreds or even thousands of containers within their ‘Micro-Services’ centralised network infrastructure. [91]

Kubernetes can be automated to dynamically scale-up more container nodes within a cluster when the application services that the container is supplying is in high demand, to provide; load balancing and rapid scalability to the application. [93]

Likewise, Kubernetes can dynamically descale a cluster when demand for the services falls, thus utilising hardware resources more efficiently and economically when containers are deployed within a CSP datacentre. [93]

Kubernetes can also monitor all containers health within a cluster to prevent an application service stopping due to a container failing, in the event of a container failure, Kubernetes will commission another container to be built from the services base image to take its place, to provide high availability and fault tolerance within a containerised cluster environment. [94]

A key feature that makes Kubernetes a very powerful tool within containerisation is the centralisation and management of persistent storage volumes that can either be stored within a CSP SAN and/or replicated to any other storage locations locally within a company or remotely within a separate CSP SAN for data redundancy which is an important feature when utilising Kubernetes with Docker. [93][91]

Kubernetes can be utilised within all the industry leading CSP within a CaaS cloud solution. AWS have also developed a proprietary containerisation management feature to CaaS called Kops [95] that is referred to as Kubernetes as a Service (KaaS) [96] that simplifies the automation process using a user-friendly management dashboard which can be purchased as additional management functionality within CaaS.

The Kubernetes control plane is constructed of 5 key components that are illustrated within figure 5.

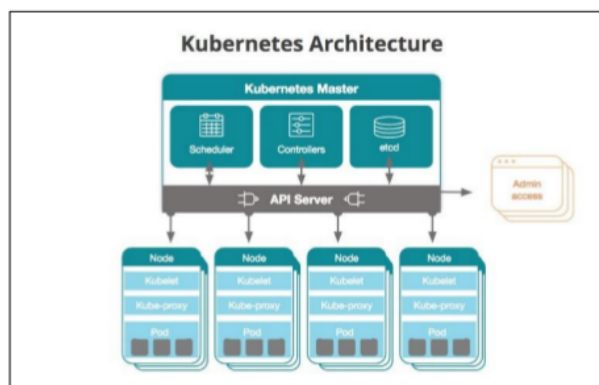


Figure 5: Kubernetes Control Plane Architecture [93]

These 5 key components and their roles within Kubernetes illustrated above have been summarised as follows: [91], [93]

1. **Kube-Apiserver:** This component exposes the Kubernetes API server at the front end of the control plane for horizontal scaling of multiple instances that can load balanced traffic between other API servers within the network.
2. **Etc Distributed (ETCD):** All cluster key value pair data is backed up and stored within this highly available persistence key store.

3. **Kube-Scheduler:** This component monitors for newly created pods within the cluster that do not have any nodes assigned and then assigns nodes to the pod based on scheduling decisions such as collective resource requirements, cluster configuration policy settings, data locality and more.
4. **Kube-Controller-Manager:** This component consists of 4 sub controllers which are; Node Monitor controller, Pod Replication Controller, Endpoint Controller, Service Default Account/API Access Token Controllers
5. **Cloud-Controller-Manager:** This component manages the sub controllers that interact with the underlying CSP infrastructure. These sub controllers are as follows; Node Deletion Controller, Route Controller, Service Load Balancing Controller and Storage Volume Controller.

A Data flow Diagram (DFD) illustrating the Kubernetes orchestration lifecycle and its components can be found in figure 6.

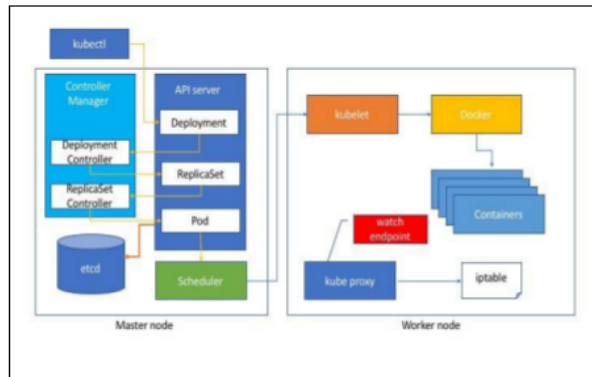


Figure 6: Kubernetes Cloud DFD Lifecycle [91]

An example of a real-world implementation of Kubernetes within industry would be a full CaaS implementation by an entertainment streaming services such as; Hulu whom have millions of customers and petabytes of video content within their platform whom store their data within a CSP datacentre. Not all of Hulu customers would be streaming video at one time i.e. each requested video file would be imbedded within a container created from the CSP CaaS image repository containing the video files stored at CSP SAN, so containers within a VPC that are not being utilised, would need to be started and stopped by an orchestration technology such as Kubernetes to efficiently manage the hardware utilisation for economic purposes along with automated scaling for when new users join or leave the service.

Load balancing due to high demand such as; in the evening after customers finish work would also be performed along with fault tolerance management to provide high availability to its customers and persistent data management of user data such as; suggested content, streaming history, payment history etc

would also need to be managed. Therefore, orchestration frameworks are vital within automating the management of containers through CaaS that by definition provides 'Micro-Services', so logic dictates that large companies such as Hulu would be utilising millions of containers every hour of each day which would be humanly impossible to manually configure and manage.

This brief research and analysis of Kubernetes container management technology, its key features and components is only a small snippet of the myriad of features and controls that are built into this powerful container orchestration technology that brings order and control to a distributed system that makes Docker and CaaS a viable alternative to PaaS and VM within the future of DevOps and business network architectures whom business technology requirements suit a 'Micro-Service' architecture.

V. CONCLUSIONS

CaaS and IaC technology are relatively new in relation to PaaS and IaaS solutions that have only been adopted by a relatively small group of businesses around the world within the last 3-5 years, due to the multi discipline knowledge that is required to create and maintain a 'Micro-Services' architecture, such as; software engineering, network and Linux CLI systems engineering skills and knowledge within the theory and implementation of orchestration technology such as Kubernetes.

A complete transition to containerisation will take many years for companies to adopt due to the small amount of multi-skilled IT professionals and the lack of knowledge within this new area of virtualisation, that will require existing network/systems/software engineers to upskill in the multi-discipline technology referred to as 'Full Stack' engineering, which would be initially very expensive for SME to implement within the short term due to the level of training required, but would be economically beneficial within the long term once non IT business professionals fully understand the true power of the technology along with the economic savings it can provide to a company.

If SME in particular are reluctant to train existing IT professionals within containerisation technologies such as Docker, Kubernetes and the Linux CLI, full adoption of containerisation technology will only take place once a new generation of multi skilled IT professionals are taught within universities or training organisations that possess the required 'Full-Stack' skills for complete adoption of a businesses IT services to a CaaS only cloud solution, and also when the Docker issues previously identified within this paper are resolved within future releases to accommodate more businesses whom have requirements that CaaS technology currently does not support.

With the advent of 5G and the Internet of Things (IoT), that are generally small networked wireless sensors that generate small

amounts of data. These devices have the perfect architecture for 'Micro-Service' and containerisation to communicate with a centralised server that could propel the CaaS CSM within the future.

VI. REFERENCES

Bibliography

- [1] Cisco, "WAN Technologies Overview (1.1) > Service Provider WAN Concepts," *ciscopress.com*, 2017. [Online]. Available: <http://www.ciscopress.com/articles/article.asp?p=2832405&seqNum=4>. [Accessed: 30-Apr-2019].
- [2] Y. Jadeja and K. Modi, "Cloud computing - Concepts, architecture and challenges," in *2012 International Conference on Computing, Electronics and Electrical Technologies, ICCEET 2012*, 2012, doi: 10.1109/ICCEET.2012.6203873.
- [3] Microsoft Azure, "What is a Cloud Service Provider – Definition | Microsoft Azure," *azure.microsoft.com*, 2019. [Online]. Available: <https://azure.microsoft.com/en-gb/overview/what-is-a-cloud-provider/>. [Accessed: 30-Apr-2019].
- [4] Amazon, "Amazon Web Services (AWS)- Cloud Computing Services," *Amazon Web Services, Inc.*, 2016. .
- [5] Microsoft, "Microsoft Azure Portal | Microsoft Azure," 2019. [Online]. Available: <https://azure.microsoft.com/en-gb/features/azure-portal/>. [Accessed: 29-Jan-2020].
- [6] Google, "Cloud Computing Services | Google Cloud," *cloud.google.com*, 2019. [Online]. Available: <https://cloud.google.com/>. [Accessed: 30-Apr-2019].
- [7] S. Chauhan *et al.*, "Amazon Virtual Private Cloud (Amazon VPC) and Networking Fundamentals," in *AWS® Certified Advanced Networking Official Study Guide*, 2018.
- [8] Issc University, "Central Processor Units (CPUs)," *Issc.edu*, vol. 1, no. 1, p. 1, 2016.
- [9] R. Wood, "Future hard disk drive systems," *J. Magn. Magn. Mater.*, 2009, doi: 10.1016/j.jmmm.2008.07.027.
- [10] U. university, "Dynamic RAM Overview," *utica.edu*, vol. 1, no. 1, p. 10, 2017.
- [11] S. Marston, Z. Li, S. Bandyopadhyay, J. Zhang, and A. Ghalsasi, "Cloud computing - The business perspective," *Decis. Support Syst.*, 2011, doi: 10.1016/j.dss.2010.12.006.
- [12] zdnet, "XaaS: Why 'everything' is now a service | ZDNet," *zdnet.com*, 2017. [Online]. Available: <https://www.zdnet.com/article/xaas-why-everything-is-now-a-service/>. [Accessed: 23-May-2018].
- [13] D. Beimborn, T. Miletzki, and S. Wenzel, "Platform as a service (PaaS)," *Bus. Inf. Syst. Eng.*, vol. 3, no. 6, pp. 381–384, 2011, doi: 10.1007/s12599-011-0183-3.
- [14] Joyent, "TRITON, Containers as a Service," *IBM.com*, 2016. [Online]. Available: <https://www.ibm.com/services/cloud/containers-as-a-service>. [Accessed: 03-Feb-2020].
- [15] Docker, "Docker overview | Docker Documentation," *docker.com*, 2018. [Online]. Available: <https://docs.docker.com/engine/docker-overview/>. [Accessed: 22-Jan-2019].
- [16] M. Artac, T. Borovssak, E. Di Nitto, M. Guerriero, and D. A. Tamburri, "DevOps: Introducing infrastructure-as-code," in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering Companion, ICSE-C 2017*, 2017, doi: 10.1109/ICSE-C.2017.162.
- [17] Cloud Native Computing Foundation, "What is Kubernetes - Kubernetes," *Kubernetes.io*, 2019.
- [18] P. Sagdeo, "APPLICATION PROGRAMMING INTERFACES AND THE STANDARDIZATION-VALUE APPROPRIATION PROBLEM," 2018.
- [19] "Software engineering frameworks for the cloud computing paradigm," *Choice Rev. Online*, 2013, doi: 10.5860/choice.51-2127.
- [20] G. Kulkarni, P. Khatawkar, and J. Gambhir, "Cloud Computing-Platform as Service," *Cloud Comput. as Serv.*, no. 2, pp. 115–120, 2011.
- [21] J. E. Smith and R. Nair, "The architecture of virtual machines," *Computer (Long. Beach. Calif.)*, 2005, doi: 10.1109/MC.2005.173.
- [22] D. Fox, "Hypervisor," *Datenschutz und Datensicherheit - DuD*, 2012, doi: 10.1007/s11623-012-0012-y.
- [23] T. Olzak, J. Boomer, R. M. Keefer, and J. Sabovik, "What is virtualization?," in *Microsoft Virtualization*, 2010.
- [24] J. Medamana and T. Siracusa, "Network functions virtualization," in *Building the Network of the Future: Getting Smarter, Faster, and More Flexible with a Software Centric Approach*, 2017.
- [25] Microsoft, "Dynamic Host Configuration Protocol (DHCP)," *microsoft.com*, 2017. [Online]. Available: [https://technet.microsoft.com/en-us/library/cc755282\(v=ws.11\).aspx](https://technet.microsoft.com/en-us/library/cc755282(v=ws.11).aspx). [Accessed: 11-Dec-2017].
- [26] ccm, "DNS (Domain Name System)," *ccm.net*, 2018. [Online]. Available: <http://ccm.net/contents/271-dns-domain-name-system>. [Accessed: 14-Feb-2018].

- [27] C. Heinle and J. Strebel, "IaaS adoption determinants in enterprises," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2010, vol. 6296 LNCS, pp. 93–104, doi: 10.1007/978-3-642-15681-6_7.
- [28] K. Brown, C. W., & Nyarko, "Software as a Service (SaaS)," *Cloud Comput. Serv. Deploy. Model. Layers Manag. Layers Manag.*, p. 50, 2012, doi: 10.1007/s11576-008-0095-0.
- [29] R. Jašek, L. Králík, and M. Popelka, "ITIL® and information security," in *AIP Conference Proceedings*, 2015, doi: 10.1063/1.4912775.
- [30] D. Beimbom, T. Miletzki, and S. Wenzel, "Platform as a service (PaaS)," *Bus. Inf. Syst. Eng.*, 2011, doi: 10.1007/s12599-011-0183-3.
- [31] "SaaS vs PaaS vs IaaS: What's The Difference and How To Choose – BMC Blogs," *BMC*, 2018. [Online]. Available: <https://www.bmc.com/blogs/saas-vs-paas-vs-iaas-whats-the-difference-and-how-to-choose/>. [Accessed: 03-Feb-2020].
- [32] P. Rotella, "Is Data The New Oil?," *Forbes*, 2012.
- [33] Redhat, "What is CaaS?," *redhat.com*, 2019. [Online]. Available: <https://www.redhat.com/en/topics/cloud-computing/what-is-caas>. [Accessed: 03-Feb-2020].
- [34] IBM, "Containers as a Service | IBM," *IBM.com*, 2019. [Online]. Available: <https://www.ibm.com/services/cloud/containers-as-a-service>. [Accessed: 03-Feb-2020].
- [35] A. Kundu and C. Banerjee, "Multi-layered service suite conceptualization in cloud computing," *J. Conver. Inf. Technol.*, vol. 5, no. 10, pp. 29–38, Dec. 2010, doi: 10.4156/jcit.vol5.issue10.5.
- [36] A. K. Yadav, M. L. Garg, and Ritika, "Docker containers versus virtual machine-based virtualization," in *Advances in Intelligent Systems and Computing*, 2019, vol. 814, pp. 141–150, doi: 10.1007/978-981-13-1501-5_12.
- [37] B. Ruan, H. Huang, S. Wu, and H. Jin, "A Performance Study of Containers in Cloud Environment," 2016, pp. 343–356.
- [38] "WHITEPAPER | ENABLING MODERN MICROSERVICES ARCHITECTURES FOR ENTERPRISE APPLICATIONS Modern Application Architecture for the Enterprise Delivering agility, portability and control with Docker Containers as a Service (CaaS)."
- [39] M. Razza, "Containers vs Virtual Machines: What's The Difference? – BMC Blogs," *BMC.com*, 2019. [Online]. Available: <https://www.bmc.com/blogs/containers-vs-virtual-machines/>. [Accessed: 03-Feb-2020].
- [40] Docker, "Docker overview | Docker Daemon Documentation," *docs.docker.com/engine*, 2019. [Online]. Available: <https://docs.docker.com/engine/docker-overview/>. [Accessed: 03-Feb-2020].
- [41] Docker, "Docker Inc," *docker.com*, 2018. [Online]. Available: <https://www.docker.com/>. [Accessed: 22-Jan-2019].
- [42] O. Delannoy, N. Emad, and S. Petiton, "Workflow global computing with YML," in *Proceedings - IEEE/ACM International Workshop on Grid Computing*, 2006, doi: 10.1109/ICGRID.2006.310994.
- [43] Docker, "Docker container networking | Docker Documentation," *Docker Inc*, 2018. [Online]. Available: <https://docs.docker.com/v17.09/engine/userguide/networking/>. [Accessed: 22-Jan-2019].
- [44] D. Namiot and M. Sneps-sneppé, "On micro-services architecture," *Int. J. Open Inf. Technol.*, 2014.
- [45] K. Jangla and K. Jangla, "Docker Compose," in *Accelerating Development Velocity Using Docker*, 2018.
- [46] S. Bistarelli, L. Kotthoff, F. Santini, and C. Taticchi, "Containerisation and dynamic frameworks in ICCMA'19," in *CEUR Workshop Proceedings*, 2018.
- [47] W. Mauerer, *Linux® Kernel Architecture*. 2008.
- [48] J. Nickoloff, *Docker in Action*. 2016.
- [49] Docker, "Repositories | Docker Documentation," *docs.docker.com*, 2019. [Online]. Available: <https://docs.docker.com/docker-hub/repos/>. [Accessed: 03-Feb-2020].
- [50] A. Brogi, D. Neri, and J. Soldani, "DockerFinder: Multi-attribute search of docker images," in *Proceedings - 2017 IEEE International Conference on Cloud Engineering, IC2E 2017*, 2017, doi: 10.1109/IC2E.2017.41.
- [51] A. Freeman and A. Freeman, "Docker Swarms," in *Essential Docker for ASP.NET Core MVC*, 2017.
- [52] IBM, "Docker Swarm vs. Kubernetes: A Comparison | IBM," *ibm.com*, 2019. [Online]. Available: <https://www.ibm.com/cloud/blog/new-builders/docker-swarm-vs-kubernetes-a-comparison>. [Accessed: 03-Feb-2020].
- [53] J. Pavlik, "The Comparison of Container Orchestrators," *ICICT2018*, 2018. [Online]. Available: https://www.researchgate.net/publication/323417485_The_Comparison_of_Container_Orchestrators. [Accessed: 03-Feb-2020].
- [54] Medium, "Kubernetes vs Docker Swarm. Who's the bigger and better?," *medium.com*, 2018. [Online]. Available:

- <https://medium.com/faun/kubernetes-vs-docker-swarm-whos-the-bigger-and-better-53bbe76b9d11>. [Accessed: 03-Feb-2020].
- [55] Hackernoon, "Kubernetes vs. Docker Swarm: A Complete Comparison Guide - By SPEC INDIA," *hackernoon.com*. [Online]. Available: <https://hackernoon.com/kubernetes-vs-docker-swarm-a-complete-comparison-guide-15ba3ac6f750>. [Accessed: 03-Feb-2020].
- [56] Mindmajix, "Kubernetes Vs. Docker Swarm : Key Facts You Need To Know," *mindmajix.com*, 2019. [Online]. Available: <https://mindmajix.com/kubernetes-vs-docker-swarm>. [Accessed: 03-Feb-2020].
- [57] "Create Hyper-V VHD Set files | Microsoft Docs." [Online]. Available: <https://docs.microsoft.com/en-us/windows-server/virtualization/hyper-v/manage/create-vhdset-file>. [Accessed: 03-Feb-2020].
- [58] A. Belak, "How to Select a Storage Approach for Persistent Containers," *Gartner*, 2017. [Online]. Available: <https://www.gartner.com/en/documents/3824065/how-to-select-a-storage-approach-for-persistent-containe>. [Accessed: 03-Feb-2020].
- [59] IBM, "Containers. Persistent storage. IBM. - IBM IT Infrastructure Blog," *IBM.com*. [Online]. Available: <https://www.ibm.com/blogs/systems/containers-persistent-storage-ibm/>. [Accessed: 03-Feb-2020].
- [60] P. Rombouts, "Container Cons: The complexities associated with Containers - SogetiLabs," *labs.sogeti.com*, 2019. [Online]. Available: <https://labs.sogeti.com/container-cons-the-complexities-associated-with-containers/>. [Accessed: 03-Feb-2020].
- [61] R. Petersen, *Linux: The Complete Reference*. 2008.
- [62] Z. Du, Z. Xu, F. Dong, and D. Shen, "A Novel Solution of Cloud Operating System Based on X11 and Docker," in *Proceedings - 5th International Conference on Advanced Cloud and Big Data, CBD 2017*, 2017, doi: 10.1109/CBD.2017.13.
- [63] C. Tozzi, "When Not to Use Docker: Understanding the Limitations of Containers," *Channel-Futures*, 2017. [Online]. Available: <https://www.channelfutures.com/open-source/when-not-to-use-docker-understanding-the-limitations-of-containers>. [Accessed: 03-Feb-2020].
- [64] Redhat, "Container Security Guide Red Hat Enterprise Linux Atomic Host 7 | Red Hat Customer Portal," *redhat.com*, 2019. [Online]. Available: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux_atomic_host/7/html-single/container_security_guide/index. [Accessed: 03-Feb-2020].
- [65] IBM, "Secure Containers - IBM," *researcher.watson.ibm.com/*, 2019. [Online]. Available: https://researcher.watson.ibm.com/researcher/view_group.php?id=9510. [Accessed: 03-Feb-2020].
- [66] D. Spinellis, "The importance of being declarative," *IEEE Softw.*, 2013, doi: 10.1109/MS.2013.18.
- [67] D. Wang, "An efficient cloud storage model for heterogeneous cloud infrastructures," in *Procedia Engineering*, 2011, vol. 23, pp. 510-515, doi: 10.1016/j.proeng.2011.11.2539.
- [68] J. Klein and D. Reynolds, "Infrastructure as Code: Final Report," 2018.
- [69] A. Rahman, R. Mahdavi-Hezaveh, and L. Williams, "A systematic mapping study of infrastructure as code research," *Inf. Softw. Technol.*, vol. 108, pp. 65-77, Apr. 2019, doi: 10.1016/j.infsof.2018.12.004.
- [70] J.-P. Klein, "Infrastructure as Code: Final Report," *resources.sei.cmu.edu*, 2018. [Online]. Available: https://resources.sei.cmu.edu/asset_files/WhitePaper/2019_019_001_539335.pdf. [Accessed: 03-Feb-2020].
- [71] A. Mikkelsen, N. Oslo, T.-M. Grønli, and R. Kazman, *Immutable Infrastructure Calls for Immutable Architecture: Deploying a Changeless Architecture in the Cloud*. .
- [72] A. Khan, "(PDF) Practicing Continuous Integration and Continuous Delivery on AWS Accelerating Software Delivery with DevOps," *Researchgate*, 2017. [Online]. Available: https://www.researchgate.net/publication/330988585_Practicing_Continuous_Integration_and_Continuous_Delivery_on_AWS_Accelerating_Software_Delivery_with_DevOps. [Accessed: 03-Feb-2020].
- [73] M. Zhang, R. Ranjan, S. Nepal, M. Menzel, and A. Haller, "A declarative recommender system for cloud infrastructure services selection," in *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 2012, doi: 10.1007/978-3-642-35194-5_8.
- [74] M. Reitblatt, M. Canini, A. Guha, and N. Foster, "FatTire: Declarative fault tolerance for software-defined networks," in *HotSDN 2013 - Proceedings of the 2013 ACM SIGCOMM Workshop on Hot Topics in Software Defined Networking*, 2013, doi: 10.1145/2491185.2491187.
- [75] M. Garrels, *Bash Guide for Beginners*. 2004.
- [76] S. V Tiponut, *Python Network Programming*. 2001.
- [77] H. Xu and D. Xiao, "Data modeling for NETCONF-based network management: XML Schema or YANG," in *International Conference on Communication Technology*

- Proceedings, ICCT, 2008, pp. 561–564, doi: 10.1109/ICCT.2008.4716122.*
- [78] M. Mark, *REST API Design Rulebook*. 2013.
- [79] H. Han *et al.*, “A RESTful approach to the management of cloud infrastructure,” in *CLOUD 2009 - 2009 IEEE International Conference on Cloud Computing*, 2009, doi: 10.1109/CLOUD.2009.68.
- [80] “(PDF) A qualitative study of major programming languages: teaching programming languages to computer science students.” [Online]. Available: https://www.researchgate.net/publication/306438454_A_qualitative_study_of_major_programming_languages_teaching_programming_languages_to_computer_science_students. [Accessed: 03-Feb-2020].
- [81] D. D6, “Deliverable D6.2 Automation and Orchestration of Services in the GEANT Community,” 2019.
- [82] Gartner, “Market Guide for Continuous Configuration Automation Tools,” *gartner.com*, 2018. [Online]. Available: <https://www.gartner.com/en/documents/3843365/market-guide-for-continuous-configuration-automation-too0>. [Accessed: 03-Feb-2020].
- [83] B. Campbell and B. Campbell, “CloudFormation-Based Tools,” in *The Definitive Guide to AWS Infrastructure Automation*, 2020.
- [84] IONOS, “Use Terraform and SaltStack for Provisioning and Configuration Management | IONOS DevOps Central,” *devops.ionos.com*, 2018. [Online]. Available: <https://devops.ionos.com/tutorials/use-terraform-and-saltstack-for-provisioning-and-configuration-management/>. [Accessed: 03-Feb-2020].
- [85] M. Zadka and M. Zadka, “Ansible,” in *DevOps in Python*, 2019.
- [86] CFEngine, “CFEngine 3.12 Documentation - Cloud Computing,” *cfengine.com*, 2019. [Online]. Available: <https://docs.cfengine.com/docs/3.12/guide-special-topics-cloud-computing.html>. [Accessed: 03-Feb-2020].
- [87] G. P. Rédei, “CHEF,” in *Encyclopedia of Genetics, Genomics, Proteomics and Informatics*, 2008.
- [88] Puppet Labs, “2015 State of DevOps Report,” *Puppetlabs*, 2015, doi: 10.1016/S0022-3913(12)00047-9.
- [89] D. Bernstein, “Containers and cloud: From LXC to docker to kubernetes,” *IEEE Cloud Comput.*, 2014, doi: 10.1109/MCC.2014.51.
- [90] J. Xu, P. Chen, L. Yang, F. Meng, and P. Wang, “LogDC: Problem Diagnosis for Declaratively-Deployed Cloud Applications with Log,” in *Proceedings - 14th IEEE International Conference on E-Business Engineering, ICEBE 2017 - Including 13th Workshop on Service-Oriented Applications, Integration and Collaboration, SOAIC 2017*, 2017, doi: 10.1109/ICEBE.2017.52.
- [91] D. Vohra, *Kubernetes Management Design Patterns*. 2017.
- [92] P. Acuña and P. Acuña, “Kubernetes,” in *Deploying Rails with Docker, Kubernetes and ECS*, 2016.
- [93] G. Sayfan, *Mastering Kubernetes*. 2017.
- [94] H. Saito, H.-C. C. Lee, and C.-Y. Wu, *DevOps with Kubernetes: accelerating software delivery with container orchestrators*. 2017.
- [95] AWS, “Manage Kubernetes Clusters on AWS Using Kops | AWS Compute Blog,” AWS, 2019. [Online]. Available: <https://aws.amazon.com/blogs/compute/kubernetes-clusters-aws-kops/>. [Accessed: 03-Feb-2020].
- [96] Rackspace, “Kubernetes-as-a-Service | Rackspace,” *rackspace.com*, 2019. [Online]. Available: <https://www.rackspace.com/managed-kubernetes>. [Accessed: 03-Feb-2020].

VII. BIBLIOGRAPHY

VIII. APPENDIX-A – DOCKER COMPOSE INDUSTRY USE CASE EXAMPLE

An example of the advantages of utilising ‘Docker-Compose’ is within an industry use case is the creating and hosting of a website where a single container could be utilised to host the webserver such as; Apache that could also comprises of the frontend of a website hosting the HTML files. This container could then be virtually networked with another container that comprises of a PHP database backend to the website. Automating and Networking container ‘Micro-Services’, is very a powerful feature within Docker because it creates a distributed lightweight software application that will provide a higher quality of service to end users, faster loading times on the client side, efficient use of hardware and ease of management and change controls for developers.

Cloud Computing Models, Containerisation Technologies and Orchestration

GRADEMARK REPORT

FINAL GRADE

92/100

GENERAL COMMENTS

Instructor

PaaS (50) 47

IaC (25) 22

K8s (25) 23

Comments

=====

Very good evaluation and comparison of PaaS vs CaaS. Although, you could have brought out the differences a bit more clearly in some parts.

Detailed discussion of IaC, could have framed the discussion more around the future benefits of this approach.

Good K8s discussion with nice example.

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11
