

Emerging and Future Trends Report

On Next-generation Cloud Computing

Module Name	Emerging and Future Trends
Module Code	ACCA7016
Lecturer	Mike Dacey / Tim Bashford
Student	Zhao Ning (1904564)
Date	03/02/2020

Table of Contents

List of Figures.....	2
Part 1. Introduction	3
Cloud Computing: Benefits and Challenges.....	3
Part 2. Docker and Containerization.....	4
2.1 Platform as A Service v. Container as A Service	4
Platform-as-a-Service (PaaS)	5
Container-as-a-Service (CaaS).....	6
Differences Between PaaS and CaaS	6
2.2 The Role of Docker	7
Advantages.....	7
Drawbacks.....	8
Part 3. Infrastructure as Code.....	8
3.1 Terminology Explained	9
3.2 Future Usage Evaluated	9
Part 4. Kubernetes.....	10
4.1 Orchestration of Containers	10
4.2 Enterprise Use Case	12
Part 5. Conclusion.....	14
References.....	15

List of Figures

FIGURE 1. VARIOUS CLOUD DELIVERY MODELS (BY MICROSOFT)	4
FIGURE 2. THE GENERAL COST HIERARCHY OF DIFFERENT CLOUD DELIVERY MODELS	5
FIGURE 3. CLOUD DELIVERY MODELS	7
FIGURE 4. AI IN DEVOPS PIPELINE (BY TARGETPROCESS).....	10
FIGURE 5. KUBERNETES ARCHITECTURE.....	12
FIGURE 6. THE ROLE OF KUBERNETES (BY MICROSOFT).....	13

CLOUD COMPUTING: BENEFITS AND CHALLENGES

Enterprises that have embraced cloud computing share some common **goals**: [1]

1). reducing costs

- Cloud computing saves enterprises the investment/cost of building, operating and maintaining their private clusters.

2). increasing scalability

- Dynamic resource pooling and allocation can be achieved in a flexible way - both through automation and manual configurations - in response to the fluctuations in demand.

3). Increasing reliability and availability

- Extensible fail-over support and highly resilient resources are usually offered through service level agreements (SLAs) to eliminate outages as much as possible and thus ensure quality-of-service.

There are also certain **risks** involved, including:

1). reduced operational control

- How cloud providers operate their cloud services is like a black box. Therefore, there will not be as much governance control over cloud assets as over on-premise assets. In the meantime, the way external connections are made - from the consumers, through the untrusted network, to the cloud servers - could pose potential risks to the enterprise.

2). increased vulnerabilities

- It is very likely that the security policies, frameworks, or controls which cloud providers support are not compatible with every single enterprise. Besides, private enterprise data stored in the cloud might be exposed to malicious attacks.

3). limited portability among cloud vendors

- Moving from one cloud provider to another could be very challenging due to a lack of established standard or authority in the cloud computing industry.

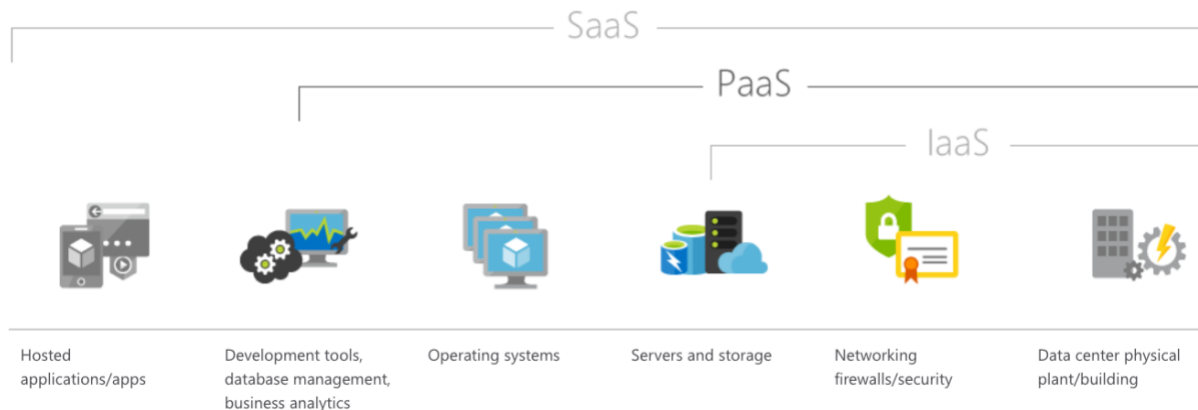
4). law and regulation compliance issues across different countries and regions

- The physical devices of large cloud providers always span more than one continent. This has raised serious legal concerns regarding data privacy and accessibility.

2.1 Platform as A Service v. Container as A Service

Various cloud delivery models exist to accommodate different organizational needs and meet specific requirements. Some of them are interrelated in how one has a scope that encompasses that of another¹, as is illustrated in Figure 1¹. One of the guidelines for organizational entities to choose the best-fit model is budget (see Figure 2²). Among these models are **Platform-as-a-Service (PaaS)** and **Container-as-a-Service (CaaS)**. Next we will give a brief introduction to both these concepts.

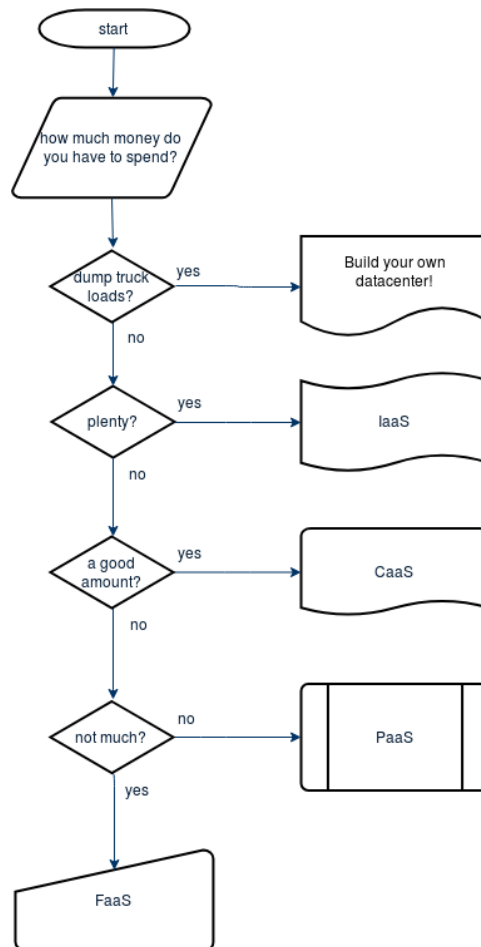
Figure 1. Various Cloud Delivery Models (By Microsoft)



¹ <https://azure.microsoft.com/en-us/overview/what-is-paas/>

² <https://developer.ibm.com/articles/when-to-use-iaas-faas-paas-and-caas/>

Figure 2. The General Cost Hierarchy of Different Cloud Delivery Models



PLATFORM-AS-A-SERVICE (PaaS)

Before introducing PaaS, we need to first gain an understanding of Infrastructure-as-a-Service (IaaS). IT infrastructures such as servers, storage, networking, and physical plants are leased and managed by cloud providers. Cloud consumers still have a high level of control over what operating system, databases, runtime environments and development tools they prefer to utilize for their systems.

Like IaaS, PaaS providers include in the services IT infrastructures like servers, storage, networking and hardware. The difference is that PaaS also includes database management systems, development tools, middleware and more features in the picture. It provides a ready-to-use environment in which pre-packaged toolkits have already been provided for the complete application delivery life cycle. For cloud consumers, this also means a decreased administrative control over the underlying IT resources.

CONTAINER-AS-A-SERVICE (CAAS)

Before introducing CaaS, we first need to know what a container is. Just like how a container in real-life scenarios would package up shipments of products, a container in cloud computing scenarios packages up everything it takes to run an application - application code, runtime environment, system libraries and other dependencies - into one lightweight, vendor-agnostic, self-contained and executable unit of software - a container image³. According the “Build once, run anywhere” concept, containerized applications will always run the same irrespective of the infrastructure. This principle is ensured by the isolation that containers create between applications and environments.

Container as a Service (CaaS) is considered by many as another cloud delivery model, right in between the IaaS model and the PaaS model. In either the on-premise data centres or the cloud environment, CaaS helps enterprises build and scale containerized applications at ease. Cloud consumers are able to upload to the cloud, start up, manage and shut down any container. Procedures like these can be conducted using APIs or web UIs⁴.

DIFFERENCES BETWEEN PAAS AND CAAS

In the case of PaaS, cloud providers are in charge of hosting, deploying, and executing programs for the consumers. Consumers have limited operational control as they rely, to a large extent, on the platform vendor’s offerings. PaaS fits better those organizations that are in a trusted relation with their cloud providers, or those that simply have next-to-none IT department to build their own solutions.

In contrast, CaaS allows consumers to have more control to modify system environments in which their applications run when needed. Therefore, in this case it is cloud customers who need to bear the operational overhead and the developers who need to obtain a clear understanding of the operation flows.

Some other popular tools present in the cloud computing landscape are listed in Figure 3⁶, including Function-as-a-Service (FaaS) which we will cover in the latter part of this report. Many other specialized models have also emerged, such as Security-as-a-Service, Database-as-a-Service and Storage-as-a-Service.

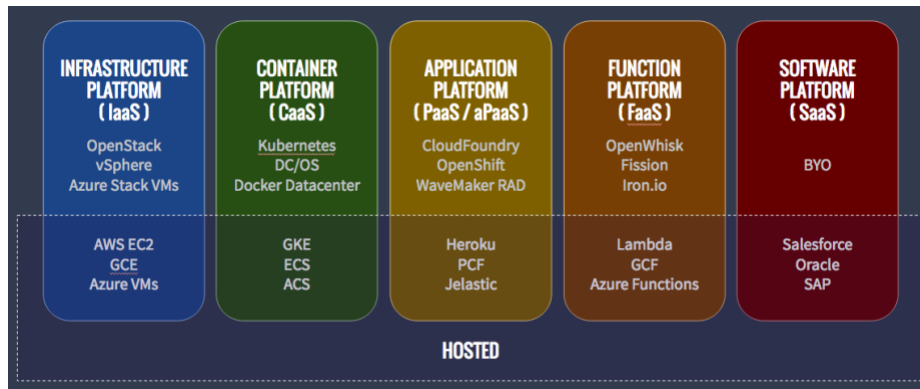
³ <https://www.docker.com/resources/what-container>

⁴ <https://www.knowledgenile.com/blogs/caas-vs-paas/>

⁵ <https://containerjournal.com/topics/container-ecosystems/caas-vs-paas-whats-right/>

⁶ <https://medium.com/@nnilesh7756/what-are-cloud-computing-services-iaas-caas-paas-faas-saas-ac0f6022d36e>

Figure 3. Cloud 7Delivery Models



2.2 The Role of Docker

Docker is the industry standard for building and sharing containerized applications. It features an agile and simple DevOps approach, and offers different choices of software development infrastructures in order for cloud consumers to deploy and deliver their applications in a flexible and quick fashion. The major components of the Docker ecosystem includes (not limited to):

- 1). **Docker Engine**: a client-server application with key features like:
 - the long-running **dockerd** command - a daemon **server** process.
 - the **docker** command - a command line interface (CLI) **client**.
- 2). **Docker Registries**
 - a Docker registry, whether private or public (Docker Hub), stores Docker images. By default, Docker is configured to search for public images published to the Docker Hub repository.
- 3). **Docker Objects**
 - including Docker images, containers, volumes, network and etc.
- 4). **Docker Services**
 - Services allow you to scale containers across multiple Docker daemons, which all work together as a swarm with multiple managers and workers.

ADVANTAGES

⁷

⁸ <https://docs.docker.com/engine/docker-overview/>

- 1). **Portability:** When an application is created in a container, that completed app has everything it needs to run, including dependencies and configuration files. This type of design provides end users with reliability and consistency when they launch applications in different environments (during the development/testing/staging/deploying phase, either locally or on public/private clouds). This also boosts the delivery process and offers organization much flexibility when it comes to choosing different cloud service providers.
- 2). **Speed:** Because containers need just enough of an operating system instead of a whole OS to run, they are super light-weight and take up much less space than virtual machines. Usually it only takes a few seconds to set up, to replicate or to shut down one of these containers. This type of design also reduces the time required to take the products to market (and thus can improve customer experience), making the releases of new versions much easier and faster.
- 3). **Scaling:** Containers enable horizontal scaling (scaling out and scaling in, i.e. the allocating and releasing of resources), allowing end users to construct (or destroy) as many containers as they wish to efficiently respond to dramatic changes in demand.
- 4). **Streamlined lifecycle:** Container-based infrastructure has an efficient software development and delivery pipeline. This elimination of environmental inconsistencies make testing and debugging much less hard and time-consuming.
- 5). **Security:** By design, containers are isolated from each other to eliminate interference among different applications. One container that has been compromised will not have a negative impact on other containers.

DRAWBACKS⁹

- 1). A new Docker version often causes breakage and unpredicted exceptions.
- 2). Docker is designed for console-based applications rather than applications with rich GUIs.
- 3). Data are created and stored inside containers by writing to container layers. Once the containers are shut down, all data stored within them will be lost forever.
- 4). Docker is not developer-friendly enough. Implementing and orchestrating large and complex applications on multiple clusters Docker containers requires quite a lot of effort.

Part 3. Infrastructure as Code

⁹ <https://www.freecodecamp.org/news/7-cases-when-not-to-use-docker/>

3.1 Terminology Explained

According to different business needs, a **continuous delivery (CD) pipeline** delivers quality products in a frequent, automatic and auditable way. Team members make a small batch of commits incrementally from testing, through quality assurance, towards production¹⁰. They used to spend a considerable amount of time, usually weeks, working with other teams involved, to provision consistent environments for different phases in the software development life cycle (SDLC). This is because the manual changes made in development, testing or staging environments are not always conveyed to the production environment. These changes can be hard to track and thus lead to errors¹¹. Over time, each environment becomes a snowflake – a unique configuration that cannot be reproduced automatically¹². This phenomenon is often referred to as **configuration drift**.

This is a situation where **infrastructure automation** turns out to be a useful solution. Also known as **infrastructure as code (IaC)**, it is one of the key **DevOps** practices, usually used in conjunction with CD. Being a descriptive model for the management of infrastructure – including storage, virtual machines and network – it replicates the same environment to up to thousands of nodes when using **scripts** to model infrastructure and its configuration.

IaC is an **agile** approach that allows both development and operations teams to apply such best practices in software development as versioning and automated testing when scripting environments at scale¹³. Applications can be tested in production-like environments early in the SDLC. It is also an **idempotent** approach in that its declarative or imperative deployment commands always configure the target environment into the same state, regardless of its beginning state or current state.

3.2 Future Usage Evaluated

DevOps provides an ideal environment for various fields of **artificial intelligence** to continue to thrive¹⁴. The accumulation of data produced in **IaC** practices can help AI analyse log reports and performance statistics, discover patterns within them, and make informed predictions in order to avoid errors that could potentially break builds by automatically carrying out appropriate

¹⁰ <https://www.atlassian.com/continuous-delivery/pipeline>

¹¹ <https://www.ibm.com/blogs/cloud-computing/2017/04/20/stop-configuration-drift-deployment/>

¹² <https://docs.microsoft.com/en-us/azure/devops/learn/what-is-infrastructure-as-code>

¹³ <https://www.ibm.com/developerworks/library/a-devops2/index.html>

¹⁴ <https://jfrog.com/blog/looking-up-4-technical-devops-predictions-for-2020/>

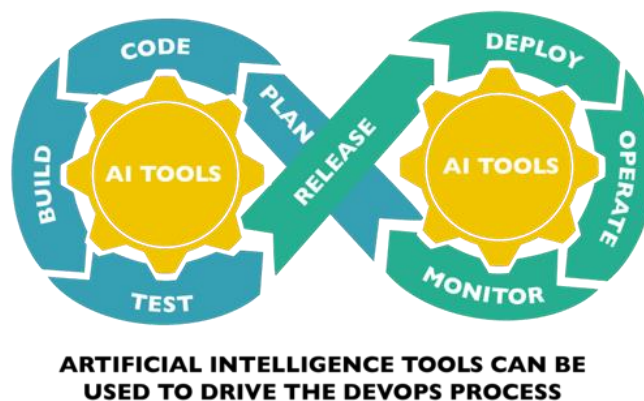
modifications to faulty code. This in turn streamlines the CI/CD pipeline, which should look like Figure 4¹⁵.

The networking industry in particular, has been supporting several initiatives towards a programmable ecosystem for network automation¹⁶.

There are four phases in a **Net-DevOps** pipeline as are shown below, in which IaC can be used during the deployment phase:

- 1) the “building” phase - where we can access blocks of pre-built configurations.
- 2) the “validating” phase - where we perform compatibility check in advance.
- 3) the “testing” phase - where we simulate various scenarios to test out the changes.
- 4) the “deploying” phase - where we can leverage API services and tools available to make agile and flexible changes in programmable network infrastructures.

Figure 4. AI in DevOps pipeline (By TargetProcess)



Part 4. Kubernetes

4.1 Orchestration of Containers

Docker is the de facto vendor-agnostic technology for packaging and distributing containerised microservices, while **Kubernetes** (K8S for short) is an open-source **orchestrator** for deploying, managing and scaling these containers created by Docker¹⁷. Often enough Docker and Kubernetes work alongside each other, allowing for container customisation not only with

¹⁵ https://www.targetprocess.com/blog/artificial-intelligence-devops-automation/#DevOps_automation_is_an_ideal_use_case_for_artificial_intelligence

¹⁶ <https://www.onug.net/blog/infrastructure-as-code-for-network-automation/>

¹⁷ <https://azure.microsoft.com/en-gb/topic/what-is-kubernetes/>

preferred programming languages, package dependencies and operating systems, but also with configurable messaging queues/buses sent among them.

K8S also provides the infrastructure needed to operate in a complex **distributed cluster** environment, which is its main design goal, as is shown in Figure 5¹⁸.

Google's Borg, a unified container-management system, is Kubernetes's **predecessor**. Borg shares cluster nodes between two types of workloads - latency-sensitive, user-facing long-running services and CPU-hungry batch jobs - as a way of boosting resource utilisation and thereby reducing costs^[2].

Compared to **Docker Swarm**, Kubernetes is more extensive and can more efficiently coordinate clusters at scale in production. Kubernetes provides high availability by distributing its basic scheduling units - pods which can contain more than one container - among cluster nodes.

K8S helps:

(1) manage **service discovery**¹⁹

Kubernetes supports two primary service discovery modes: environment variables and DNS.

- When a **pod** (described by the PodSpec YAML/JSON object) starts running on a node, the **kubelet** (the node agent) sets a series of environment variables for each running service.
- DNS service can be set up for any Kubernetes cluster using an add-on like the cluster-aware DNS server CoreDNS.

(2) incorporate **load balancers**²⁰

- In Kubernetes, an Ingress controller makes use of existing load balancing algorithms and weight schemes for backend servers to help manage the traffic.

(3) **track** the allocation of computing resources and **scale** accordingly²¹

- In Kubernetes, a horizontal pod auto-scaler, which specifies resource metrics the kube-controller-manager later queries the resource utilization against, will change the number of active pod replicas to maintain an average CPU utilization across all active pods of 50%.

(4) **monitor** the **health** of microservices

- Kubernetes offers readiness/liveness/health check using HTTP endpoints or exec commands that users specify to run within pods^[3]. It further enables self-healing to

¹⁸ <https://medium.com/containermind/a-beginners-guide-to-kubernetes-7e8ca56420b6>

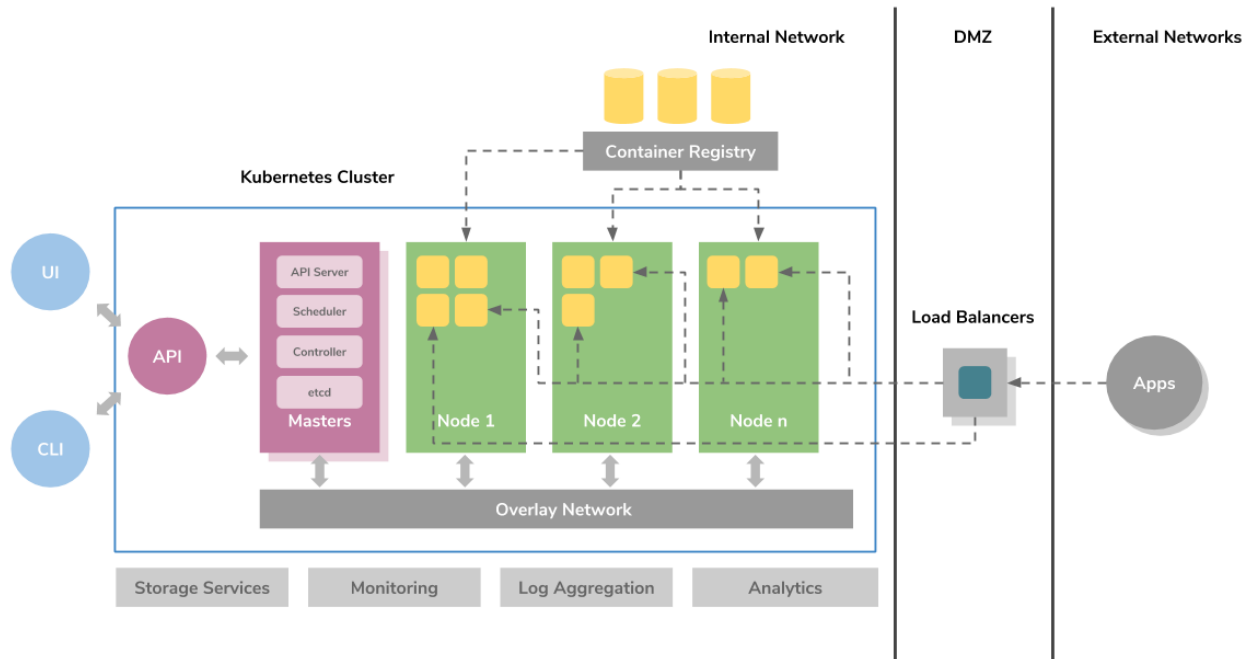
¹⁹ <https://kubernetes.io/docs/concepts/services-networking/service/#discovering-services>

²⁰ <https://kubernetes.io/docs/concepts/services-networking/ingress/>

²¹ <https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

improve reliability in distributed systems by restarting containers that fail, and replacing or killing those that are not responding²².

Figure 5. Kubernetes Architecture

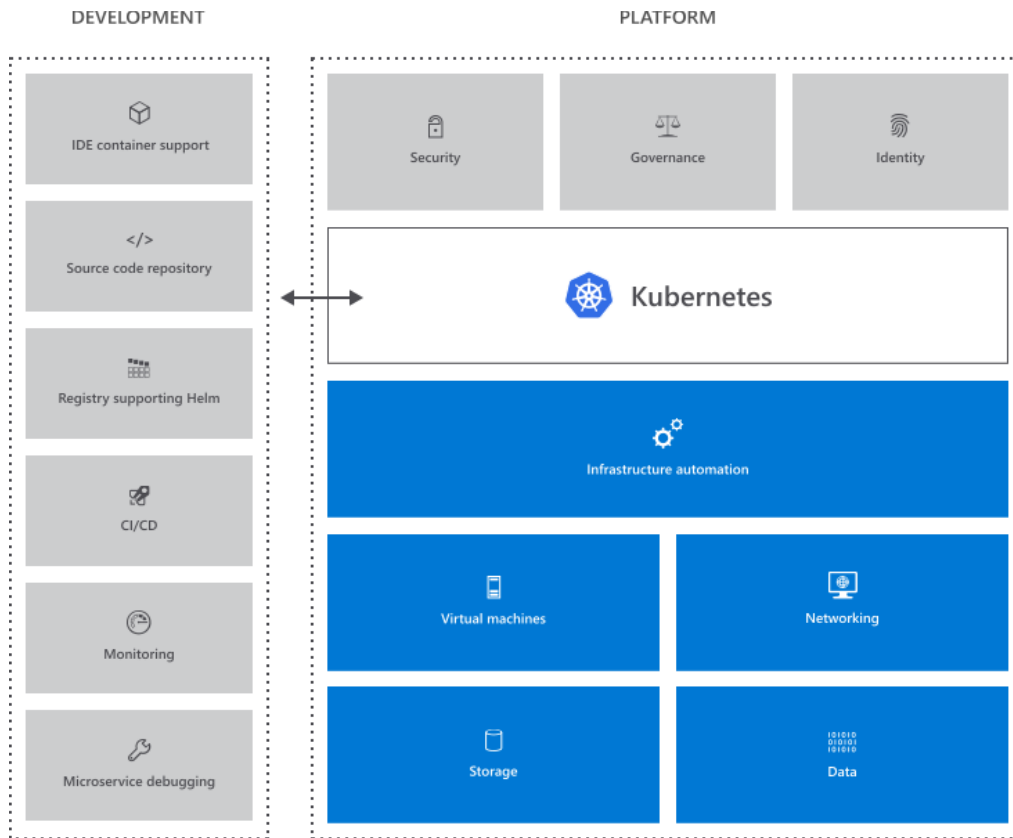


4.2 Enterprise Use Case

K8S can integrate with existing **IaC** and **CI/CD** tools to schedule and deploy project releases., as is shown in Figure 6.

²² <https://azure.microsoft.com/en-us/topic/what-is-kubernetes/>

Figure 6. The Role of Kubernetes (By Microsoft)



In the age of Everything-As-A-Service²³, Kubernetes provides core features to build container-management APIs, eliminates infrastructure lock-in²⁴, and shifts the focus of data centres from machines to applications[2].

Here we will present a use case from Ant Financial, a spin-off of the multinational conglomerate Alibaba, to demonstrate how Kubernetes has solved the challenges it faced²⁵. Ant Financial handles a staggering volume of transactions for up to thousands of millions of users worldwide, with its peak reaching more than 200,000 per second on its annual 11/11-special-celebration day. Operating this kind of business scale, its system cannot afford to have a service downtime for even seconds, otherwise there would be a huge negative impact on its businesses and consumers both in and out of China.

After investigating several orchestration solutions for the large number of clusters in its data centres, it chose Kubernetes eventually because of its vibrant and fast-growing open-source

²³ <https://blog.container-solutions.com/when-is-the-wrong-time-to-use-kubernetes>

²⁴ <https://www.infoworld.com/article/3173266/4-reasons-you-should-use-kubernetes.html>

²⁵ <https://kubernetes.io/case-studies/ant-financial/>

community. And it later turned out that this cloud native technology has enabled Ant Financial to scale globally and operate efficiently²⁶.

However, orchestration of containers is not the end and Kubernetes cannot be a one-size-fit-all solution. The emerging paradigm of **serverless computing**, in which applications' computing resources are managed by third-party vendors (e.g., Function-as-a-Service platforms like Cloudflare Workers²⁷), could be more cost-efficient than containers. Containers keep running and thus causing expenses even when there is not a single request for their services. In contrast, serverless applications cease to exist and thus drive costs to zero, unless they are explicitly accessed²⁸. Serverless computing also allows for scaling without the complexity and headaches of managing containers.

Google's Cloud Run, for example, is a unique end-to-end managed cloud computing platform that takes Docker containers and produces a URL instantly. From SSL provisioning at the top, all the way down to running the container, it fully and automatically handles the scaling of stateless containers. Platform service subscribers pay on a pay-as-you-use basis, only when the code is running²⁹.

Part 5. Conclusion

Cloud computing will become the norm in the near future, with enterprises assembling their legacy applications and a mixture of cloud services (hybrid clouds or multi-clouds) to accommodate their specific needs.

Besides, the benefits that open-source cloud computing technologies bring will continue to attract enterprises to migrate to the cloud: interoperability across different cloud environments, protection from vendor lock-in, and continuous innovation contributed by developers from all around the world³⁰. The impressive progress of these game-changing technologies like Docker containers and Kubernetes orchestrator is and will continue to be made possible by the open-source world.

²⁶ <https://kubernetes.io/case-studies/ant-financial/>

²⁷ https://www.cloudflare.com/learning/serverless/what-is-serverless/?utm_referrer=https://www.google.com/

²⁸ <https://thenewstack.io/why-serverless-vs-kubernetes-isnt-a-real-debate/>

²⁹ <https://techcrunch.com/2019/04/09/google-cloud-run-brings-serverless-and-containers-together/>

³⁰ <https://services.google.com/fh/files/misc/futurecloudcomputing.pdf>

References

- 1 Erl, T., Puttini, R., and Mahmood, Z.: 'Cloud Computing: Concepts, Technology & Architecture' (Prentice Hall Press, 2013. 2013)
- 2 Burns, B., Grant, B., Oppenheimer, D., Brewer, E., and Wilkes, J.: 'Borg, Omega, and Kubernetes', 2016, 14, (1), pp. 70-93
- 3 Hightower, K., Beda, J., and Burns, B.: 'Kubernetes: Up and Running: Dive Into the Future of Infrastructure' (O'Reilly Media, 2017. 2017)