Integration of UNIX-Based Network Services with Cloud Computing Platforms

Kai Siang Kao (103819380)

Abstract—**This report provides a comprehensive exploration of the deployment of UNIX and non-UNIX systems in cloud environments, offering insights into crucial considerations for organizations seeking optimal performance, flexibility, and cost-effectiveness. It highlights the major cloud service providers, including AWS, Microsoft Azure, and Google Cloud Platform, as well as various tools available to deploy them, comparing their implications and use cases. The differences in executing UNIX systems on PaaS/CaaS and IaaS environments are also discussed, with insights into the respective management responsibilities.**

I. Introduction

Cloud computing has rapidly emerged as a prevailing paradigm within the technological landscape. Presently, approximately 50% of organizational data has found its home within cloud infrastructures, signifying a significant shift in data storage and management practices. Projections indicate that by the year 2025, an estimated half of the world's data will have transitioned to reside within the cloud [1].

This transition has been facilitated by the abundant flexibility offered by cloud service providers, catering to a diverse spectrum of user profiles. From casual hobbyists to proficient prosumers, and extending to business owners and decision-makers, cloud services have proven adaptable to various utilization scenarios. Concurrently, a multitude of operating systems are available on the market, with an array of options presented by cloud providers themselves.

In light of this transformative landscape, the criticality of selecting the optimal cloud platform aligns with the growing complexity of these offerings. It is essential that the alignment of specific cloud capabilities with corresponding usage scenarios to ensure optimal outcomes.

II. Background

Before moving to the analysis, it is crucial to understand some of the key concepts that will be discussed.

A. Operating Systems

An operating system (OS) is a software application that, following its initial loading into a computer by a boot program, assumes control over all other programs within the computer. These programs typically consist of application software that interact with the operating system by sending service requests through a well-defined Application Programming Interface (API) [2].

B. UNIX

Unix, legally trademarked as UNIX, represents a multiuser and multitasking operating system (OS) renowned for its adaptability and versatility. Originating in the 1970s, Unix emerged as a pioneer among operating systems that were constructed using the C programming language. Following its inception, the Unix OS and its derivatives exerted a significant influence on the computer and electronics sector, delivering traits such as mobility, reliability, and harmonious functionality across diverse environments and device categories. Unix notably stood as the OS that welcomed contributions for enhancements and refinements from a wide array of individuals, partly due to its utilization of the C language and its embrace of numerous popular concepts [3].

However, its early success led to multiple variants that lacked compatibility and interoperability. As such, a standard has been as a solution to standardize all the Unix-based operating systems in the market. The POSIX standard was born. It stands for Portable Operating System Interface and is developed and maintained by the IEEE Computer Society's Portable Application Standards Committee (PASC) [4]. This standard outlines a collection of functions that results from blending elements of AT&T UNIX System V and Berkeley Standard Distribution UNIX. All systems adhering to the POSIX standard are obligated to incorporate these functions, and software aligned with the POSIX standard exclusively employs these functions to access services from both the operating system and the foundational hardware. Adherence to the POSIX guidelines streamlines the process of migrating applications from one POSIX-compliant operating system to another [5].

C. Examples of UNIX and non-UNIX operating system

Various criteria exist for delineating what falls within the realm of UNIX and what does not. Although the term UNIX initially denoted a specific operating system developed by AT&T Bell Research Labs in the 1960s, its contemporary interpretation encompasses operating systems that derive from UNIX. As previously mentioned, a standard exists that provides UNIX certification upon an operating system, thereby granting it the privilege to employ the UNIX® trademark. However, only a select few companies, including Apple, IBM, and HP, have chosen to pursue this certification [6]. The majority of Linux distributions do not possess UNIX certification due to the intricacies and costs associated with the certification process. Nonetheless, a significant portion of

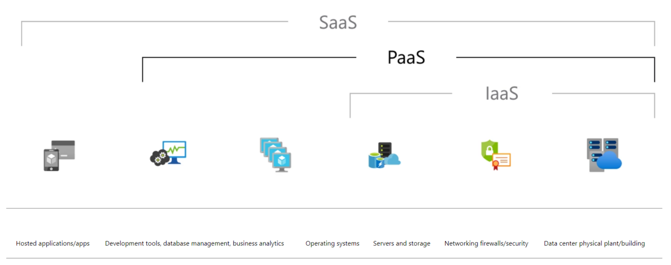
SaaS
PaaS
IaaS
Hosted applications/apps
Development tools, database management, business analytics
Operating systems
Servers and storage
Networking firewalls/security
Data center physical plant/building

Fig. 1. Different level of abstractions of different as-a-service models [11]

these distributions maintains adherence to the POSIX standard [7].

In the context of this report, Linux distributions will be classified within the Unix classification. Conversely, the primary focus of this report will revolve around Windows as a prominent example of a non-Unix operating system. Although there exist other operating systems that diverge from the Unix framework, such as ReactOS (which also operates within the Windows realm) [8], their utilization is limited, typically catering to individuals with specific and targeted intentions. For such users, the distinction of whether an operating system is Unix-based or not holds little significance.

III. Cloud

According to the definition provided by NIST [9], "*Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics, three service models, and four deployment models.*".

Luis M. Vaquero et al. [10] addressed the inherent challenges in definitively defining the concept of cloud computing due to its expansive nature. They highlighted a pivotal role for cloud service providers, emphasizing their responsibility to render services accessible to Service Users through web-based interfaces. These services are delivered as a comprehensive infrastructure provided by the service providers. The authors listed three primary usage scenarios for these cloud services: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS).

With the popularity of cloud computing, more models have been introduced to the landscape, expanding the range of available options. These models encompass diverse categories, such as Backend as a Service (BaaS), such as platforms like Firebase and Supabase; Database as a Service (DBaaS), such as PlanetScale and MongoDB Atlas; and Storage as a Service (STaaS), such as Google Cloud Storage and AWS S3.

However, within the context of the present paper, numerous services are abstracted and managed by the service providers themselves. Consequently, certain as-a-service models are excluded from consideration. The primary focus lies on the as-a-service paradigms that necessitate the selection of operating systems, mainly Infrastructure as a Service (IaaS) and Container as a Service (CaaS).

IV. Advantages and Disadvantages of Deploying UNIX on Cloud

In the context of cloud computing, users with distinct profiles would have different performance metrics and requirements in a cloud system. For instance, individuals like hobbyists or students often seek user-friendly cloud solutions, while businesses prioritize attributes such as stability, scalability, and security within their cloud environments. These divergent requirements underscore the necessity for tailored cloud services that cater to a spectrum of user needs.

Akinlolu Adekotujo et al. [12] provided a comprehensive overview of different operating systems, highlighting on their unique characteristics and trade-offs. According to the paper, Windows distinguishes itself by its high user-friendliness and strong reliability. However, its susceptibility to significant security threats is a notable concern. Geared towards a broad audience and commercial applications, Windows systems are renowned for offering robust technical support, albeit at a premium price point. The trade-off between accessibility and security is highlighted, as the widespread user base of Windows systems makes them attractive targets for hackers. Additionally, the graphical user interface of Windows contributes to its slower performance in comparison to Unix as a tradeoff of a more user-friendly environment.

In contrast, Unix stands out for its balanced combination of user-friendliness, moderate security vulnerabilities, and heightened reliability. Unix's remarkable customizability presents both advantages and disadvantages. While granting users extensive control over system configurations enhances compatibility with individual preferences, it also entails the risk of disrupting system integrity. Unix also necessitates the presence of an expert for even routine tasks like installing new products or applying updates.

V. Integrating UNIX-Based Networking Services on the Cloud

Similarly to Windows Server, Unix possesses the capability to offer a variety of networking services. For instance, it can utilize dnsmasq for DNS and DHCP functions [13], vsftpd (Very Secure File Transfer Protocol Daemon) for FTP services [14], Apache for handling HTTP requests [15], along with a multitude of other options. Nevertheless, there exist several crucial factors to contemplate when incorporating these networking services within Unix-based cloud systems.

A. Security

Security is a very important aspect when running a machine on a cloud. While UNIX is generally considered a more secure operating system (at least compared to Windows), this section will present some aspects to consider in terms of security in a cloud environment.

1) DDoS: Denial of Service (DoS) attacks are carried out to obstruct legitimate users' access to cloud networks, storage, data, and other services. These attacks typically involve compromising a service that can extensively utilize cloud resources such as computational power, memory, and network bandwidth. This disrupts cloud operations, leading to operational delays and renders the cloud unresponsive to other legitimate users and services. Distributed Denial of Service (DDoS) attacks represent a variant of DOS attacks, employing multiple network sources to inundate the cloud with an overwhelming volume of requests, depleting its resources. These attacks exploit vulnerabilities in web servers, databases, and applications, ultimately resulting in resource unavailability [16]. During the integration of networking services, there's a susceptibility to DoS attacks, potentially leading to the interruption of active services. Furthermore, in a cloud environment, falling victim to a DoS attack could result in substantial financial expenses, as increased computing resources are allocated to process these unsuccessful requests that offer no benefit to the cloud user.

2) Shared Technology Vulnerability: Cloud computing introduces a multi-tenancy system where various users utilize diverse cloud resources. However, the foundational elements that constitute the infrastructure for deploying cloud services might not have been initially designed to ensure robust isolation properties suitable for accommodating a multi-tenant framework or applications serving multiple customers. This situation can give rise to shared vulnerabilities across the technology stack, encompassing components like Virtual Machines (VMs), operating systems, and hypervisors. A vulnerability or misconfiguration within a shared platform component could grant an attacker the means to compromise the security of cloud data for numerous or all customers, resulting in a breach of data. Implementing cautious approaches in client setup and data management serves as a safeguard against the vulnerabilities inherent in shared technology environments [17]. Maintaining these practices becomes especially vital due to the challenge of detecting these vulnerabilities by cloud users, given that the cloud provider abstracts them from direct visibility. Malicious individuals capable of circumventing the confined virtual environment can gain entry to the machine. Employing a robust client setup can effectively reduce the damage done by such occurrences.

3) Firewall and IDS: Chaimau Saadi et al. [18] highlighted the limitations of operating a traditional firewall in a cloud environment. Only a few efforts have been made towards cloud firewalls. One solution proposed is a centralized cloud firewall, but physical security devices like firewalls and Intrusion Detection Systems (IDS) without prevention mechanisms continue to face challenges. The seriousness of threats remains high. Traditional Detection Systems also struggle to understand alerts effectively.

To enhance security against internal and external attacks, the authors proposed a more robust architecture which includes a decentralized cloud firewall for safeguarding user tenants and applications hosted in the cloud infrastructure. Additionally, a Host-based Intrusion Detection and Prevention System (IDS/IPS) monitors incoming traffic to identify malicious activity. A correlation strategy aids in better understanding alerts.

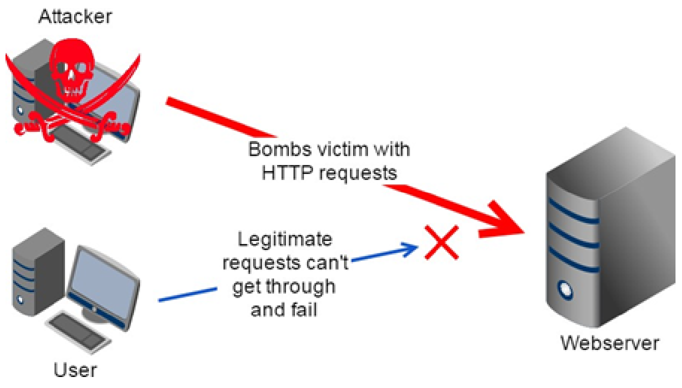
Attacker
Bombs victim with HTTP requests
Legitimate requests can't get through and fail
User
Webserver

Fig. 2. Denial of Service Attack [17]

Cloud service providers offer the cloud firewall, situated between the internet and the cloud data center. Customers can rent this firewall to protect their tenants and applications within the cluster. Firewall resources are allocated dynamically to set up individual firewalls for each cluster. These parallel firewalls work together to monitor incoming packets and reinforce security measures in the cloud environment.

Deploying an IDS/IPS is also crucial to protect all virtual machines against different attacks, the main goal of this system is to identify and remove intrusions in real time. The authors also gave a few examples of open-source HIDS (Host-Based Intrusion Detection System) that are compatible with Linux/UNIX systems, such as OSSEC, Tripwire, AIDE, and SAMHAIN.

B. Monitoring and Logging

1) Logging: Raffael Marty [19] conducted a study on cloud application logging for forensics. Firstly, it is shown that logging serves several valuable purposes, including debugging, forensics, fault monitoring, troubleshooting, feature usage tracking, performance monitoring, incident detection, and regulatory compliance. Cloud-based applications generate logs across multiple servers and log files, but due to the volatile nature of resources, log files are available for a limited time. Each layer within a cloud application stack contributes to log generation, encompassing the network, operating system, applications, databases, and network services. In the context of UNIX systems, syslog is often pre-configured for logging in the `/var/log` directory. However, configuring and tuning OS logs, such as logging password changes, can be challenging. Application-level logging can be lacking or misconfigured, leading to missing log records and essential information.

The author proposed a guideline of when and what to log. More specifically, determining when to generate log records is driven by use-cases in cloud applications, categorized into business relevance, operational events, security-related incidents, and regulatory compliance. A general guideline is to log status information at every return call within an application, ensuring errors are recorded and activities throughout the application can be tracked. Essential fields for each log record include Timestamp, Application, User, Session ID, Severity, Reason, and Categorization. These fields address questions related to when, what, who, and why, supplying necessary information for various use-cases. Ensuring these fields are present facilitates effective log analysis and fulfills the demands of different logging scenarios.

2) Monitoring: Yong Peng et al. [20] classified the technology into two primary categories: hardware-based and software-based. Hardware-based monitoring utilizes specialized hardware devices to track hardware events. For instance, processors incorporate performance counters to tally cache misses. In contrast, software-based monitoring captures and quantifies events arising from software execution, with support available from either software or hardware. An example of this is monitoring the frequency of context switching in the operating system.

According to a study conducted by Giuseppe Aceto et al [21], monitoring in cloud environments is influenced by several critical factors. These factors encompass the need to engage in capacity and resource planning for estimating computing resources and workload projections. Additionally, monitoring plays a pivotal role in ensuring continuous uptime, efficient data center management, compliance with service level agreements (SLAs), accurate billing, troubleshooting, performance optimization, and security management. The authors also pointed out that Hyperic-HQ as a cloud monitoring tool compatible for UNIX systems.

C. Cost Management

Although several strategies for reducing costs can be employed in cloud environments, not all of them are viable for network service operations. For instance, AWS EC2 and Microsoft Azure provide spot instances as a cost-effective alternative to their normal VM instances [22] [23], but this approach is unsuitable due to the imperative need for network services to maintain exceptionally high availability levels.

1) Auto-Scaling: Fabio Pianese et al. [24], have proposed the development of a UNIX-based operating system explicitly tailored for cloud computing, with a versatile design to accommodate diverse cloud use cases. Their Cloud OS places a strong emphasis on achieving a higher level of integration with network resources. This Cloud OS is essentially a collection of distributed processes whose primary objective is resource management, offering the following key characteristics:

- It extends its functionality to grant administrative access to Cloud resources, allowing tasks such as virtual machine allocation and deallocation, process dispatching and migration, and the establishment of inter-process communication channels.
- It provides a set of network-based interfaces that applications can utilize to interact with the management system and exert control over Cloud resources.

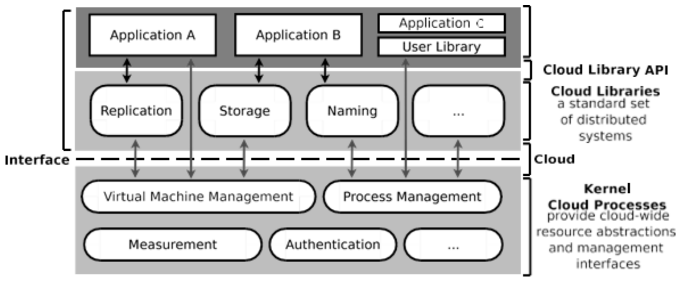

Fig. 3. Logical model of Cloud OS as proposed [24]

- It incorporates software support for autonomously adjusting the scale and deploying distributed applications opportunistically within the Cloud environment.

Figure 3 shows a Cloud OS conceptual model proposed by the authors, defining a Cloud object as a collection of local OS processes on one node, each with a unique identifier to prevent system-wide ID conflicts. Cloud processes (CPs) include Cloud objects for distributed applications. The Cloud architecture distinguishes kernel space CPs for resource management and user space CPs for running User Applications and Cloud Libraries. Applications communicate with these components via network-based standard interfaces, Cloud System Calls. The architecture is hardware-flexible, necessitating Cloud kernel processes and trust credentials for node participation. Cloud user space objects offer signal handling and network-based management. The naming Library manages object-to-network mappings, the authentication kernel CP handles access rights, and Measurement kernel CPs operate as needed [24].

Implementing an auto-scaling cloud infrastructure is highly valuable for cost management. Given the varying userbase demands throughout different periods, it is impractical to reserve capacity for peak traffic levels year-round when it mostly remains underutilized. Auto-scaling enables the deactivation of idle resources, effectively achieving cost savings as a result.

VI. Tools Available for Running UNIX on Cloud

This section discusses some of the tools available for running UNIX on Cloud.

A. Virtualization

Numerous tools exist for virtualizing operating systems, and one notable example is VirtualBox. It is a robust x86 virtualization solution caters to both enterprise and personal use, and it is available as Open Source Software under the GNU General Public License (GPL) version 3. VirtualBox boasts compatibility with a wide range of operating systems, encompassing Windows, Linux, macOS, OpenBSD, and more. It maintains an active development cycle with frequent updates [25]. However, it's essential to note that while VirtualBox serves as a valuable tool for testing UNIX operating systems within a controlled environment, deploying nested virtualization is generally discouraged due to the cloud environment's inherent layer of virtualization [26].

TABLE I
COMPARISON OF DIFFERENT TOOLS AVAILABLE

Category	Virtualization	Containers	Cloud Orchestration	Cloud Services
Tools Available	• VirtualBox	• Docker	• Kubernetes	• Amazon Elastic Compute Cloud (EC2) • Azure Virtual Machines • Google Compute Engine (GCE)
Advantages	• Suitable for local testing • Lightweight	• Lightweight • Standardized deployment environment	• Allows ease of management with containers • Scalable	• Pay-as-you-go model • Highly scalable
Disadvantages	• Not recommended to be deployed on cloud	• May not be suitable for all type of services	• Steep learning curve, especially for beginners • Requires more setup and configuration	• Requires a lot of management • May introduce vendor lock-in

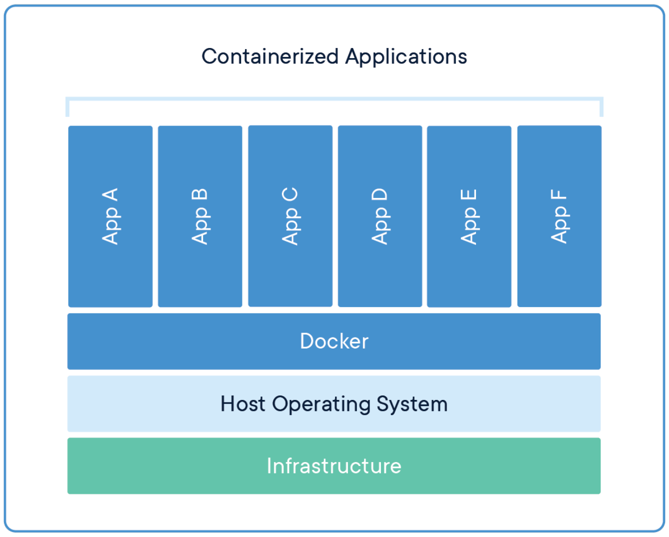

Fig. 4. Underlying structure of a Docker container [27]

B. Containerization and Orchestration

A container represents a standardized software unit that encapsulates code and all its prerequisites, allowing an application to execute consistently and efficiently across various computing environments. Docker is one of the tools that offer this functionality. Specifically, a Docker container image serves as a lightweight, self-contained, executable software package encompassing all elements necessary for application execution: code, runtime, system tools, system libraries, and configurations. These container images transform into operational containers when executed, particularly in the case of Docker containers, which become active when run on the Docker Engine. Containerized software is compatible with both Linux and Windows-based applications, ensuring consistent performance across diverse infrastructures. [27].

Running UNIX in a container offers a notable advantage: it allows individuals using different operating systems on their machines, each equipped with diverse software and versions, to establish a uniform environment for application execution. This standardization helps circumvent compatibility problems associated with mismatched software setups. DockerHub hosts a vast repository of Docker images containing various applications and configurations designed for multiple operating systems, including UNIX, making it convenient for users to access and deploy pre-configured environments.

While containers are an effective means to package and run applications, but in a production environment, managing these containers and ensuring uninterrupted operation becomes crucial. Kubernetes [28] is a tool that helps with managing containerized workloads and services. It furnishes a robust framework for running distributed systems with resilience and handles critical tasks like scaling, failover management, deployment strategies, and more. Key features include service discovery and load balancing, storage orchestration, automated rollouts and rollbacks, automatic bin packing for resource optimization, and self-healing capabilities. Additionally, Kubernetes offers secret and configuration management, allowing the secure storage and management of sensitive data, all without requiring container image rebuilds or exposing secrets in stack configurations. Nonetheless, it's essential to remember that Kubernetes differs from a traditional Platform as a Service (PaaS) system and lacks a complete suite that enables independent operation.

C. Cloud Service Providers

A multitude of cloud service providers exists in the market, but the dominant trio comprises Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP), collectively commanding 66% of the global cloud infrastructure market [30]. These providers own the data centers where virtualized machines are hosted, offering a similar array of

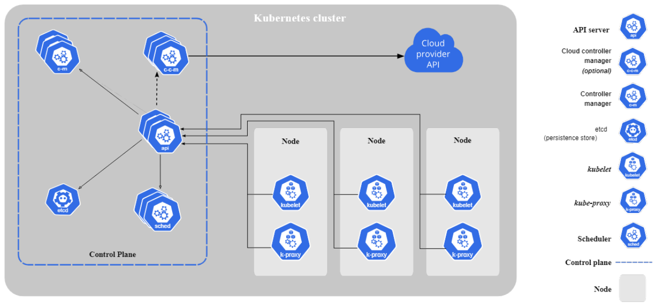

Fig. 5. Components of a Kubernetes cluster [29]

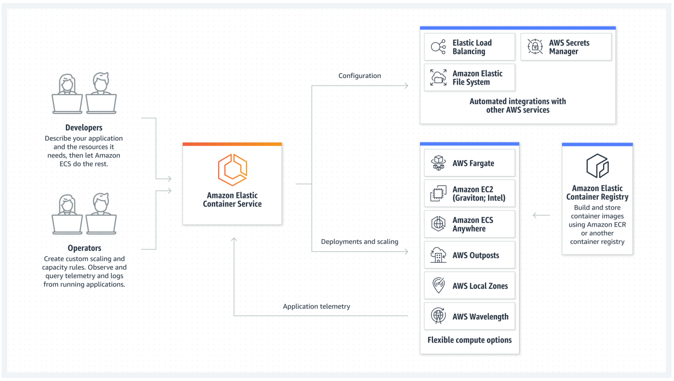
Developers
Describe your application and the resources it needs, then let Amazon ECS do the rest.
Operators
Create custom scaling and capacity rules. Observe and query telemetry and logs from running applications.
Amazon Elastic Container Service
Configuration
Deployments and scaling
Application telemetry
Elastic Load Balancing
AWS Secrets Manager
Amazon Elastic File System
Automated integrations with other AWS services
AWS Fargate
Amazon EC2 (Graviton; Intel)
Amazon ECS Anywhere
AWS Outposts
AWS Local Zones
AWS Wavelength
Flexible compute options
Amazon Elastic Container Registry
Build and store container images using Amazon ECR or another container registry

Fig. 6. Deployment Architecture of AWS ECS [33]

services that cater to a wide range of use cases. In the context of deploying UNIX systems in the cloud, beyond the containerized environment discussed in Section VI-B, UNIX can also be installed on a VM rented from the provider, denoted as Infrastructure as a Service (IaaS). This approach grants users full control above the infrastructure layer, leveraging UNIX's modularity and customizability. The synergy between UNIX and IaaS provides users unparalleled flexibility in tailoring their environment to specific needs, thereby minimizing unneeded or bloat services and leading to cost-effective solutions.

VII. UNIX as CaaS/PaaS vs IaaS

As outlined in Section VI, UNIX systems can operate effectively on both Container-as-a-Service (CaaS) and Infrastructure-as-a-Service (IaaS) platforms. In the case of Platform-as-a-Service (PaaS), its underlying operating system is abstracted away from the user by cloud providers, bearing little significance to users, provided the platform aligns with the hosted application. Nevertheless, several PaaS providers, such as AWS Elastic Beanstalk [31] and Heroku [32], is based on Linux, likely due to its performance advantages and its suitability for server environments.

Running UNIX within a container diverges significantly from its role as the primary operating system. While containers like Docker can define an application's runtime environment, they are dependent on a host operating system for execution, rather than a hypervisor. This lightweight approach leverages the host operating system's kernel but necessitates hosting on top of an operating system. For example, AWS's Container-as-a-Service (CaaS), Amazon Elastic Container Service (ECS) [33], offers automated scaling of applications across various compute options. Consequently, a compute option must still be provided, either via a virtual machine like EC2 or a serverless platform like Fargate. Alternatively, Kubernetes serves as a cloud orchestration tool, facilitating container deployments with seamless integration into cloud service providers.

Conversely, deploying UNIX on Infrastructure-as-a-Service (IaaS) allows the operating system to function on top a hypervisor. Here, the guest operating system operates as an independent entity, with resource allocation capabilities. Additionally, any compatible operating system can be installed, as long as it is supported by the hypervisor. However, this additional flexibility also necessitates comprehensive management by the cloud user, such as security, resource allocation, and service configuration, extending beyond the hardware layer.

VIII. Conclusion

In conclusion, this report has highlighted the critical distinctions between deploying Unix and non-Unix systems in the cloud. Particularly for businesses reliant on these services, the choice of the most suitable system is especially crucial. Failing to meticulously weigh the pros and cons may lead to the accumulation of technical debt, potentially resulting in substantial financial and time-related setbacks when the time comes to finally fix it.

References

[1] S. Morgan. "The World Will Store 200 Zettabytes Of Data By 2025," Cybercrime Magazine. (Jun. 2020), [Online]. Available: https://cybersecurityventures.com/the-world-will-store-200-zettabytes-of-data-by-2025/ (visited on 09/01/2023).
[2] E. Walia, *Operating system concepts*. KHANNA PUBLISHING HOUSE, 2002.
[3] R. Sheldon and E. Mixon. "Unix," Tech Target. (2023), [Online]. Available: https://www.techtarget.com/searchdatacenter/definition/Unix (visited on 09/01/2023).
[4] A. Josey. "POSIX™ 1003.1 Frequently Asked Questions (FAQ Version 1.18)," The Open Group. (Jun. 2020), [Online]. Available: https://www.opengroup.org/austin/papers/posix_faq.html (visited on 09/01/2023).
[5] D. Lewine, *POSIX programmers guide*. " O'Reilly Media, Inc.", 1991.
[6] T. O. Group. "UNIX® Certified Products," The Open Group. (May 2023), [Online]. Available: https://www.opengroup.org/openbrand/register/ (visited on 09/01/2023).
[7] phuclv. "Is there a Linux distro that's UNIX certified?" (Oct. 2018), [Online]. Available: https://unix.stackexchange.com/a/476945 (visited on 09/01/2023).
[8] ReactOS. "FAQ — ReactOS Project." (2023), [Online]. Available: https://reactos.org/faq/ (visited on 09/01/2023).

[9] P. Mell, T. Grance, *et al.*, “The nist definition of cloud computing,” 2011.

[10] L. M. Vaquero, L. Rodero-Merino, J. Caceres, and M. Lindner, *A break in the clouds: Towards a cloud definition*, 2008.

[11] Microsoft. “What is PaaS?” (2023), [Online]. Available: https://azure.microsoft.com/en-au/resources/cloud-computing-dictionary/what-is-paas (visited on 09/02/2023).

[12] A. Adekotujo, A. Odumabo, A. Adedokun, and O. Aiyeniko, “A comparative study of operating systems: Case of windows, unix, linux, mac, android and ios,” *International Journal of Computer Applications*, vol. 176, no. 39, pp. 16–23, 2020.

[13] Simon Kelly. “dnsmasq - Unix, Linux Command.” (n.d.), [Online]. Available: https://www.tutorialspoint.com/unix_commands/dnsmasq.htm (visited on 09/01/2023).

[14] Chris Evans. “vsftpd - Secure, fast FTP server for UNIX-like systems.” (2021), [Online]. Available: https://security.appspot.com/vsftpd.html (visited on 09/01/2023).

[15] Apache. “Welcome! - The Apache HTTP Server Project.” (2023), [Online]. Available: https://httpd.apache.org/ (visited on 09/01/2023).

[16] M. Kazim and S. Y. Zhu, “A survey on top security threats in cloud computing,” *International Journal of Advanced Computer Science and Applications*, vol. 6, no. 3, pp. 109–113, 2015.

[17] P. Suryateja, “Threats and vulnerabilities of cloud computing: A reviews,” *International Journal of Computer Sciences and Engineering*, vol. 6, no. 3, pp. 297–302, 2018.

[18] C. Saadi and H. Chaoui, “A new approach to mitigate security threats in cloud environment,” in *Proceedings of the Second International Conference on Internet of things, Data and Cloud Computing*, 2017, pp. 1–7.

[19] R. Marty, “Cloud application logging for forensics,” in *proceedings of the 2011 ACM Symposium on Applied Computing*, 2011, pp. 178–184.

[20] Y. Peng and I.-C. Wu, “A cloud-based monitoring system for performance analysis in iot industry,” *The Journal of Supercomputing*, vol. 77, pp. 9266–9289, 2021.

[21] G. Aceto, A. Botta, W. De Donato, and A. Pescapè, “Cloud monitoring: A survey,” *Computer Networks*, vol. 57, no. 9, pp. 2093–2115, 2013.

[22] Amazon Web Services. “Spot Instances.” (2023), [Online]. Available: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-spot-instances.html (visited on 09/01/2023).

[23] Microsoft. “Azure Spot Virtual Machines.” (2023), [Online]. Available: https://azure.microsoft.com/en-au/products/virtual-machines/spot (visited on 09/01/2023).

[24] F. Pianese, P. Bosch, A. Duminuco, N. Janssens, T. Stathopoulos, and M. Steiner, “Toward a cloud operating system,” in *2010 IEEE/IFIP Network Operations and Management Symposium Workshops*, IEEE, 2010, pp. 335–342.

[25] Oracle. “Welcome to VirtualBox.org!” (2023), [Online]. Available: https://www.virtualbox.org/ (visited on 09/01/2023).

[26] Saturn Cloud. “Can VirtualBox be Executed Under an Amazon EC2 Instance?” (Aug. 2023), [Online]. Available: https://saturncloud.io/blog/can-virtualbox-be-executed-under-an-amazon-ec2-instance/ (visited on 09/01/2023).

[27] Docker. “Use containers to Build, Share and Run your applications.” (2023), [Online]. Available: https://www.docker.com/resources/what-container/ (visited on 09/01/2023).

[28] Kubernetes. “Overview — Kubernetes.” (Jul. 2023), [Online]. Available: https://kubernetes.io/docs/concepts/overview/ (visited on 09/01/2023).

[29] Kubernetes. “Kubernetes Components — Kubernetes.” (Jul. 2023), [Online]. Available: https://kubernetes.io/docs/concepts/overview/components/ (visited on 09/01/2023).

[30] Synergy Research Group. “Cloud Spending Growth Rate Slows But Q4 Still Up By $10 Billion from 2021; Microsoft Gains Market Share.” (Feb. 2023), [Online]. Available: https://www.srgresearch.com/articles/cloud-spending-growth-rate-slows-but-q4-still-up-by-10-billion-from-2021-microsoft-gains-market-share (visited on 09/01/2023).

[31] Amazon Web Services. “Elastic Beanstalk Linux platforms.” (2023), [Online]. Available: https://docs.aws.amazon.com/elasticbeanstalk/latest/dg/platforms-linux.html (visited on 09/02/2023).

[32] Heroku. “Heroku Dynos - Build Process — Heroku,” SalesForce. (2023), [Online]. Available: https://www.heroku.com/dynos/build (visited on 09/02/2023).

[33] Amazon Web Services. “Amazon Elastic Container Service.” (2023), [Online]. Available: https://aws.amazon.com/ecs/ (visited on 09/02/2023).