Business model innovation using modern DevOps

DVSR Krishna Koilada
NetrixLLC
Houston, Texas, USA
kkoilada@netrixllc.com

***Abstract*— The paper aims at evaluating modern DevOps tools, strategizing the execution models and monetizing the business model innovation. DevOps, with its pragmatic tools, processes, and solutions, is in the early adoption stage of the technology life cycle. At this technology innovation stage, the adopting Organizations benefit the most from the inception to execution. However, not all business firms, depending on the operating mode in the market, experience the tangential outcomes. Despite the limitations, from a technology perspective, DevOps fosters collaboration among diverse teams and eliminates cultural disparities. As a result, high-performing technology organizations deliver applications or solutions with reduced lead-times. Changing the view scope with a different lens, DevOps appears a new entrepreneurial opportunity for technology and business leaders. Combining these two seemingly disparate perspectives and invigorating the appeal strategy to the executive management for business model innovation is the core of the subject; the new frontier for influencing the consensus on technology adoption and hence, the experiential innovation for transforming business models.**

Keywords- DevOps, Business models, Containers, Orchestration, Micro-services, B2B, B2C, Technology Architecture

I. INTRODUCTION

DevOps stands for development and operations. Conventional and current industry practices are such that these two business operational segments work independently with little crossroads for the collaboration. The objective of DevOps principles is to bridge the gaps among functional groups such as IT Operations, application developers, Information Security and formulate a new cultural environment. As shown in figure 1.1, the key strategy is to fuse technology, processes, and culture, transforming the team dynamics. One of the core principles to achieve higher velocity stresses on automation, curbing or eliminating any manual operations. Through the advanced technology-enabled automation tools and processes, the cloud service providers and privately-owned Data-center firms are banking on new business offerings with the adoption of modern DevOps tools. Sometimes Agile and lean methodologies are mistakenly viewed as DevOps implementation; however, it is meant merely for faster software development. Agile enables the strategic and tactical framework of application software development, whereas DevOps focuses on both development and delivery segments. Combining Agile, lean and DevOps practices, the industry verticals enhance the throughput and reduce lead-times [1] establishing the collaboration, transparency, and trust among Developers, QA and Operations as shown in fig 1.2

Utilizing modern DevOps technologies and principles, organizations are fostering toward high-performers category with rapid continuous-delivery and continuous-deployments of applications. The time-gap factor between releases and deployments qualifies whether the organization is in the high-performer zone. Besides, the collaboration activities, irrespective of geographical boundaries, among different functional groups play a vital role in performer qualifiers zone. However, modern tools such as Dockers and containers are landscaping the architecture of the cloud-native application. These tools are offering the technology industries to revamp organizational metrics through the measure of delivery, reliability and availability modes of the applications and services, which are provided over the cloud platforms.

Tasked at minimizing the gaps between Continuous Integration (CI) and Continuous Delivery and Deployments (CD) [2], DevOps technology framework aims at achieving effective build pipelines. The typical journey of a build pipelines starts at the coding development stage, fusing the source code and testing the code. Also included system integration tests, this journey segment is termed as continuous integration. One of the strategies of modern DevOps architecture is effective implementation of test-driven development (TDD), and the tactical execution of TDD [3] is the utilization of automation tools, eliminating any human intervention. Essentially, TDD environment enforces the test cases development prior to actual functional programming. This framework enables organizations to achieve high-velocity by burning down tasks efficiently and become a high-performer [4] in the competitive technology market space. Upon achieving the defined criteria for the test cases to pass, the integrated code begins its journey to the next stage of the build pipeline. At this stage, the code enters into artifact repository which classified as Continuous Delivery (CD), transitioning into release management life cycle. Either manual or automated approval process, the release is set to travel its final journey of the pipeline stage, i.e., continuous deployment in the production environments.

978-1-7281-1139-1/19/$31.00 ©2019 IEEE

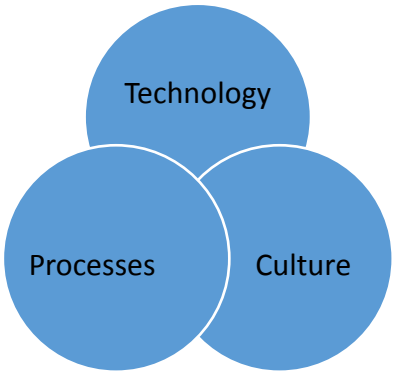

Fig 1.1: DevOps Strategy

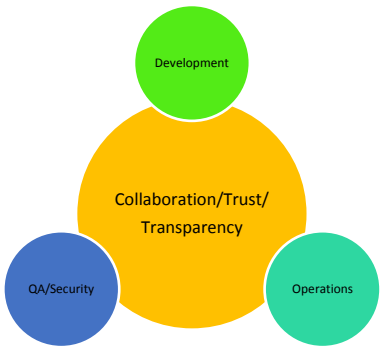

Fig 1.2: DevOps Tactics

II. Technology Architecture

With the exponential technology advancements over time, the virtualization has transformed the high-performance computing platform. Moreover, the evolved changes have restructured the application and solution delivery platform fabric. Virtualization is considered the mothership for all the new rapid developments which in other words has enabled the growth of cloud platforms. Transforming the physical hardware resources to a virtual segment, Operation systems (OS), as shown in fig 2.1, which host the business applications, are no longer confined to utilize the physical or bare-metal computing hardware in its entirety. Hypervisor [5] which sits on top of bare metal hardware is a software that extracts and virtualizes the compute resources. There are two types of Hypervisors namely Type-1 and Type-2, and the difference between them lies in the installation segment. Classified to handle enterprise and business application loads and higher performances, Type-1 runs directly on bare metal hardware, whereas Type-2 runs on the Guest/Conventional OS that virtualizes and abstracts the compute resources assigned to OS.

Either with Type-1 or Type -2 deployments, any number of OS instances can spawn with oversubscription feature enabled on the resource utilization fabric. Accordingly, the applications are developed and delivered on these virtual machines. So far this has been the proven most effective usage of the computing resources. However, each instance of VM does not allow running the multiple instances a.k.a user spaces of the same application services.

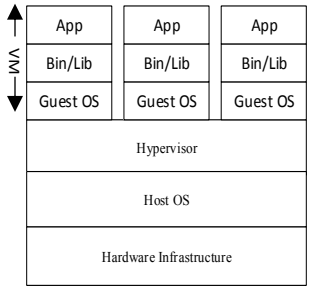

Fig 2.1: Virtual Machine

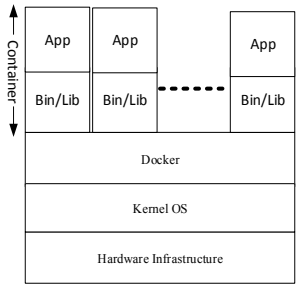

Fig 2.2: OS virtualization

Also, IT operations become complex running and managing the gamut range of each application instances. It is even posing challenges on operational and capital expenses for both service providers and customers where IT organizations are confined to operate with no viable routes for bringing in cost savings. These limitations have led to break-through development of another layer of virtualization, i.e., OS-level virtualization. OS-level virtualization, as shown in fig 2.2, allows the virtual machine to run multiple isolated application user-spaces instances in parallel.

1) Docker and Container Architecture

Docker [6] is an open source technology which enables segregation of applications from the infrastructure. Three essential components lay the foundation for the docker framework namely a) Registry b) Docker host and c) Docker client. From an application standpoint, Docker requires binaries, dependencies and set of libraries for the application code functioning. The packaging of the code and dependencies termed as image or docker image. Usually, the smaller the image, the faster the deployment. So, the images are advised to keep it minimal, recommending the segmentation of associated services with separate images. In doing so, the applications' communication transforms to micro-services docker architecture. Moreover, once the image is built, it is then pushed to the docker registry which holds the images facilitating as images repository system.

In contrast, the Docker client and most often the docker host(server) run together on the same OS user space and communicate via API [7]. Docker daemon which lifts heavy workloads preparing the application runtime communicates to the registry for pulling and building the appropriate app image(s). Besides, the docker daemon containing the application image does the groundwork for instantiating user space on the OS, allocating the process id(s) and associated network and storage mount points. Finally, the docker client runs the instance of the app's docker image. The container host containing multiple instances of isolated images is referred to as containers, and a running instance is called a container runtime. Essentially, Docker is a platform which hosts all containers or container runtime. In concise, these user-spaces contain the application code, required libraries, and the runtime process to start an application. And,

these instances are available through the package of application code, dependencies, and libraries. The container is created from an image and runs as a process on the OS kernel. As a result, on the host OS, one can run or spin multiple containers from the same image. And, the OS's kernel assumes responsibility isolating and providing the resources to each container runtime.

2) *Orchestration Architecture*

It is highly unlikely that one would be interested in the involvement of tedious manual and time-consuming process for running the container applications. Hence, the container orchestration tools will come into play to achieve the deployment scalability, administering and managing the multiple container runtimes in any given infrastructure [8] environments either small, medium or large. Fortunately, there are various orchestration tools available in the marketplace both commercial and open-source. Two popular orchestration tools widely in use are a) Docker Swarm b) Kubernetes. Although Docker Swarm is easy to install, the limitations such as lack of auto-scaling, automatic rollbacks, reliance on 3rd party tools for monitoring, GUI supporting higher demands, have stifled the tool adoption on a large scale by the practitioners.

In contrast, Kubernetes [9], an open-source orchestration container software which is widely adopted and utilized by all industry-leading cloud service providers, offers much more flexibility for application scalability and deployments. Initially developed by Google and contributed to cloud-native computing foundation (CNCF), Kubernetes is an orchestration tool to administer, manage and automate the availability of the container runtimes reliably and scale or shrink the container pods as needed. Pods are a logical abstraction of the desired configuration for the container runtimes. Kubernetes works as a Master and worker node architecture model. The master node running in a single or multi-node mode supports user interactions via API and facilitates scheduling and controlling in addition to maintaining a key-value (KV) store for cluster-configurations.

On the other hand, the worker node, which runs on the docker node, hosts kubelet, an agent responsible for deploying container run-times. Both master and worker nodes communicate at all times securely via the API server, ensuring the container pods are running at the desired state. With the help of replication controllers, the orchestration tool safeguards the application availability at any given point in time. Therefore, replication controllers ensure that the container run times are healthy for any given pods.

The current practices allow each virtual instance or a machine to run different application and services on each OS. For scaling the applications horizontally, a similar instance is spin up and configured to meet the growing demands. Combining the set of services and running them on a single virtual machine is vertical scalability of the application. However, the scalability of the service is limited in this vertical expansion segment, no matter in the increase of the computing resources due to I/O operations within a host OS kernel. This limitation has been counterintuitively overcome with the horizontal scaling of the application where multiple virtual host machines are configured to run the same application, load sharing the service requests and operations. Interacted through the network stack and application load balancer, service-oriented architecture (SOA) drives the platform availability and resiliency. In this architecture mode, the I/O operations are associated with North-South bound traffic, i.e., the application either sends or receive the requests flowing from top layer fabric. However, the case is different for container runtime applications. The container and its orchestration fabric transform the serviceability of the applications through micro-services architecture.

In this mode, each container image is configured to run the desired service or application, effectively breaking the services into individual segments. In doing so, the micro-service architecture has emerged. Contrary to North-South bound I/O service request and responses in SOA, Micro-services architecture heavily involves in East-West bound I/O operations on top of North-South data flows due to the underlying fabric of the container and docker communications. Containers, Dockers and their orchestration are the modern technology trends, evangelizing the modern DevOps practices. As a result, this technology architecture, which classifies as a modern DevOps segment, reinvigorates the availability, reachability and reliability modes of the application and services.

III. Business Model Architecture

The modernization of the technology architecture offers the businesses to explore and land on the new opportunities. Irrespective of the market and organizational size, cloud service providers can monetize this modern DevOps technology innovation. Therefore, the value-based proposition is more tangential in the multi-tenancy application or services model. In this context, the segmentation of the customers is done through the application interface layer. Yet, the same application running on the virtual instance is utilized by all the customers.

Moreover, the underlying application code and services are shared among multiple customers. Small and medium businesses that are less concerned for sharing the application with other tenants are focused on the solutions and services segments that fulfill their needs. However, some companies operating in the compliance or regulatorily mandated divisions require data segregation, privacy, and security isolation. Those needs are achieved at a premium cost for the organizations since the applications are now made to run separate virtual instances. In doing so, results in additional operational and support costs for IT operations as well as for end customers.

Leveraging the modern DevOps technology architecture, the businesses embark on new offerings, as shown in fig 3.1, to the customers' dynamics. The typical multi-tenant X-as-a-service applications can be offered as an exclusive X-Platform-as-a-service (XPaaS) for those market segments or customers that need segregation of the data transactions. Moreover, exercising the new technology-

enabled architectures, the businesses can target enterprise customers, expanding the reach from small-medium business (SMB) to small-medium-enterprise (SME) segments. Besides, the business leaders have a strategic advantage penetrating unexplored existing and new market segments. Furthermore, the tactical execution of XPaaS business model enables a strategy the service providers vitalize on the new business model innovation and the associated offerings. Technology management leadership in collaboration with business stakeholders can now upsell the technology offerings as a new business model to existing or prospective customers, creating a win-win scenario for both businesses and customers.

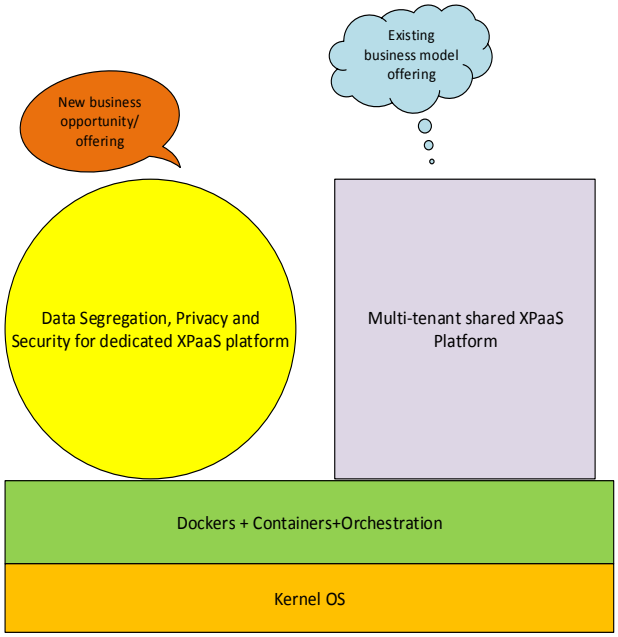

Fig 3.1: Strategic business model offering

IV. Considerations

Although the modern DevOps offers many value-added propositions to various industry verticals, the adoption and practice of container orchestration architecture are not suited for all phases of organizational life-cycle stages, as shown in fig 4.1, especially for startups and early stage of startups. Startups operate in a context trying to achieve a product-market fit and promoting the growth of the product. The early growth stage of the startup requires active customer engagement at all times, reducing the lead-times in the delivery [10] of application and feature enhancements. Modern DevOps implementation requires different skill sets and trained workforce personnel in contrast to existing non-container-based technology architecture. Engaging the resources on this new learning curve requires a significant amount of time as well as training costs. Unfortunately, startups do not have flexibility or freedom to afford the expenses or time on a segment that is not aligned with their vision or mission. The similar logic applies to startups of early growth stage as well. However, once the startups have achieved the product-market fit [11] and growth with some gross profit revenues, channelizing the efforts on new technology architecture is justified and recommended before it becomes a complex undertaking over time with transitional delays. In doing so, the scalability and growth of the products or services synchronize with the customer and market demands, and hence the organizational growth and competitiveness remain intact.

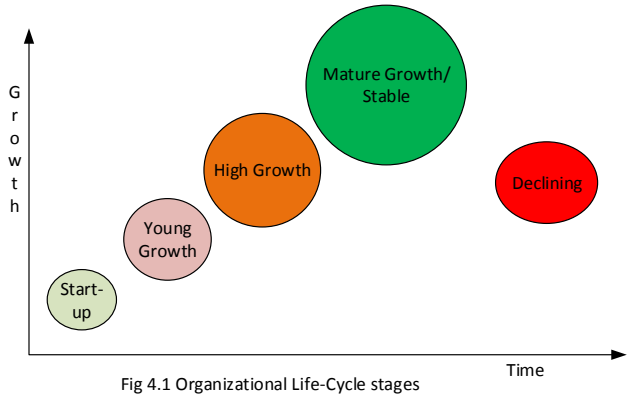

Fig 4.1 Organizational Life-Cycle stages

V. Case Study: DevOps transformed Multi-tenancy cloud unified communications and systems

Multi-tenancy cloud Unified Communications systems provide the customers with a modern Telephony platform for exchanging the productivity tasks involving human voice interactions. VoIP is the most widely used protocol for enabling communications among the end devices. The media traffic flows underneath the communication channels take a heavy toll on I/O computing resources. Also, the multi-tenancy business model enables service providers to onboarding multiple customers on the same application platform. However, the segregation of the customers is at the user interface level. In doing so, the business model becomes cost-effective for both organizations and customers. In the context of the organizational lifecycle, multi-tenancy systems business model can be positioned anywhere in the growth curve, i.e., a startup to decline stage, depending on the nature of the application and its business model. Although profitable, multi-tenancy does not provide similar cost savings for the customer segments that need data segregation, privacy and security. For meeting these particular needs, the applications are deployed on the separate instance of virtual machines architecture contributing to the company's additional CAPEX and OPEX.

Enabling the service-oriented architecture (SOA) [12] for the VoIP and media applications has posed challenges in terms of reliability and availability. Since the application dependencies are running on the same virtual machine that hosts the business application, any failure in one of the dependency segments makes the app unusable. These limitations are effectively solved with micro-services enabled DevOps technology architecture, pushing the availability and reliability metrics to the highest possible level. Once the business attains or reaches a high growth [13] stage of the organization's life-cycle plot, implementing the DevOps technology architecture is rightly justified. At this

stage, the revenues are flowing in but with a positive operating profit margin. The reason for justifying the actions is based on the business's operating committee ability to invest financial resources, not at the expense of negative operating margin, for hiring and training technical personnel. In contrary, bringing such changes at early startup or young growth stage may result in dire consequences due to negative operating cash flow.

To illustrate further on the case study, let's revisit the multi-tenancy Unified Communications systems architecture. The product and its services are offered over the cloud with user segregation taking place at the application interaction layer. Primarily, multiple clients are using the same application infrastructure that is shared with other tenants. Hence, the onboard customers' jobs-to-be-done segments do not conflict with the existing offering models. As it is not desirable to run a heavy load with more tenants boarded on the same shared infrastructure, the conventional method is to spin up new virtual machines for scaling the application horizontally. Furthermore, the vertical scalability of any application has its limits with performance issues bubbling up on the surface. So, at the expense of burning additional VM stack resources, the scalability is achieved with an increase in operations' complexity.

To achieve the growth of the application and ease the operational challenges, DevOps' infrastructure technology is effectively strategized by running the unified communications applications on the platform that utilizes Docker, Container and Orchestration tools, promulgating the growth of the business. Precisely, the same software build pipelines used for multi-tenancy environment are leveraged to offer the services as a separate segregated dedicated environments. From an entrepreneurial scope, for example, the value proposition is streamlined with the offerings that the client's communication services are completely isolated, providing security and data segmentation along with fault-tolerant architecture. As a consequence, the customers have been influenced buying on that value proposition as they experience the unique value with these offerings. In effect, the business model is restructured with DevOps's architecture innovation, channelizing the offerings not only to the direct clients/customers in a B2C context but also to channel partners in a B2B context. Had the offering been made without the modern DevOps architecture, the firm's ability to provide segmented services and run the operations at positive operating profit margin would be in question.

By transforming the architecture, not only the technology-management leadership has improved the operating environment performance but also the business offering segment has evolved with an opportunity to entertain the customers of any segments with data segregation, privacy, and security. Effectively, this business model innovation offers freedom and flexibility to the end customers choosing the platform fabric according to their needs. In consequence, the business influences and upsell the new offerings to existing and prospective customers.

VI. Conclusion

Modernizing the technology architecture and strategizing the DevOps frameworks, technology organizations discover a new way of solving the limitations and transform the operating environment. Modern DevOps practices evangelize automation for all segments, eliminating human involvement and enhancing the performance of the organizations. In the technology space, these tools shape the architecture of continuous delivery and deployments, providing a new key performance indicator (KPI) for measuring technology success. While the technology trend is in the early adoption stage, the businesses strategize on product and service differentiation and growth of the organization with new ventures or offerings in the untapped market segments before the market place gets densely crowded.

References

[1] Gene Kim, Jez Humble, Patrick Debois, & John Willis (2016) "The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations",pp.8-12.

[2] Sanjeev Sharma (2017) "The DevOps Adoption Playbook: A Guide to Adopting DevOps in a Multi-Speed IT Enterprise": Wiley IBM Press,pp.106-141

[3] Sriram Chandrasekaran, Sauri Gudlavalleti, and Sanjay Kaniyar,(July 2014) "Achieving success in large, complex software projects". Retrieved fromhttps://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/achieving-success-in-large-complex-software-projects

[4] Nicole Forsgren PhD, Jez Humble, Gene Kim (2018) "Accelerate: The Science of Lean Software and DevOps: Building and Scaling High Performing Technology Organizations", p. 36

[5] Stephen J. Bigelow (Jan 2019) "What's the difference between Type 1 and Type 2 hypervisors?". Retrieved from https://searchservervirtualization.techtarget.com/feature/Whats-the-difference-between-Type-1-and-Type-2-hypervisors

[6] Docker architecture, Retrieved from https://docs.docker.com/engine/docker-overview/#docker-architecture on April 17, 2019

[7] Aruna Ravichandran, Kieran Taylor, Peter Waterhouse (2016) "DevOps for Digital Leaders: Reignite Business with a Modern DevOps-Enabled Software Factory":CA Press, pp. 51-67

[8] Thomas Delaet and Ling Lau (March 28, 2017)"DevOps:The key to IT infrastructure agility". Retrieved from: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/digital-blog/devops-the-key-to-it-infrastructure-agility

[9] Kubernetes architecture, Retrieved from https://kubernetes.io/docs/concepts/overview/components/ on April 11 ,2 019

[10] Oliver Bossert, Chris Ip, and Irina Starikova (September 2015) "Beyond agile: Reorganizing IT for faster software delivery". Retrieved from: https://www.mckinsey.com/business-functions/digital-mckinsey/our-insights/beyond-agile-reorganizing-it-for-faster-software-delivery

[11] Eric Ries (2011), "The Lean Startup", pp. 219-221

[12] Ross Altma, Kirk Knoernschild (July 2014) "SOA and Application Architecture Key Initiative Overview", Retrieved from https://www.gartner.com/doc/2799817/soa-application-architecture-key-initiative#a-1119860124 on April 17,2019

[13] Aswath Domadaran, "THE CORPORATE LIFE CYCLE: GROWING UP IS HARD TO DO!" Retrieved from http://people.stern.nyu.edu/adamodar/pdfiles/country/corporatelifecycleLongX.pdf on April 19, 2019

DVSR Krishna Koilada is pursuing EMBA at HAAS School of Business from University of California, Berkeley, CA, USA and received the Master's (MS) degree in Technical Management from Johns Hopkins University, Baltimore, MD, USA and the Master's (MS) degree in Telecommunication from University of Louisiana, Lafayette, LA, USA. His expertise spans in the areas of Unified Communications, Telecommunications, Systems Architecture, Carrier services, Hybrid clouds, Product Management and Product leadership. His interests are in start-ups, tech-entrepreneurship, technology and innovation management and organizational management.