International Journal of Innovative Research in Computerand Communication Engineering

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

Deploying and Exposing a React JS Application in Azure Kubernetes Service

Divyenth K, Naveen Raj V, Vignesh U, Vignesh Kumar A

B. E Student, Dept. of.CSE, Bannari Amman Institute of Technology, Sathyamangalam, India

B. E Student, Dept. of.CSE, Bannari Amman Institute of Technology, Sathyamangalam, India

B. E Student, Dept. of.CSE, Bannari Amman Institute of Technology, Sathyamangalam, India

Assistant Professor Level II, Dept. of.CSE, Bannari Amman Institute of Technology, Sathyamangalam, India

ABSTRACT: To develop and build a modern cloud applications or DevOps implementation, both Docker and Kubernetes have revolutionized the era of software development and operations. Although both are unique, they unify the process of development and integration; it is now possible to build any architecture by using these technologies. Docker is used to build, ship and run any application anywhere. Docker allows the use of the same available resources. These containers can be used to make deployments much faster. Containers use less space, are reliable and are very fast. Kubernetes is an automated container orchestration tool that helps in managing containers, deployment and scaling platform. Using Azure Cloud Platform to deploy containers on Kubernetes Engine enables rapid application development and management. Kubernetes provides key features like deployment, easy ways to scale, and monitoring, rolling updates and much more. All these processes can be automated into the CI/CD pipeline using tools like Jenkins and Jenkins x which will further reduce the overhead of the development and operations team.

KEYWORDS: Docker and Kubernetes; Azure Kubernetes Service; React JS; CI/CD Pipeline

I.INTRODUCTION

Docker and kubernetes have taken the era of cloud computing by storm. This paper discuses about how kubernetes and docker is used to deploy applications in a very short time and how applications on the cloud have 0 downtime, how application updates are performed and how the CI/CD pipeline can be automated using tools like Jenkins and Jenkins x. Traditionally in order to deploy any application on the cloud VMs were used. But the drawback was that it was not possible to deploy multiple applications with different dependencies on the same VM. And another major problem was that applications were not working the same in all environments. It was also difficult to maintain configurations in case multiple micro services were used. There may be several micro services that build one complete application in such cases these micro services or containers must work in harmony and in a well-organized manner where one container can talk to another. Kubernetes comes into play to achieve such an environment, kubernetes is a container orchestration platform and is supported by almost all cloud providers. This paper focuses about AKS which is Azure Kubernetes Service nothing but Kubernetes running in Azure. Once application is deployed in AKS and exposed it can be accessed across the globe and the application is always up and running that is no or zero downtime all this may seem like a lengthy process but once all this is automated in the CI/CD pipeline all this can be done with a single command.

II.DOCKER AND CONTAINER ORCHESTRATION

A. DOCKER:

Docker is a set of platforms as a service (PaaS) products that uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their own software, libraries and configuration files; they can communicate with each other through well-defined channels. All containers are run by a single operating-system kernel and are thus more lightweight than virtual machines [1].

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

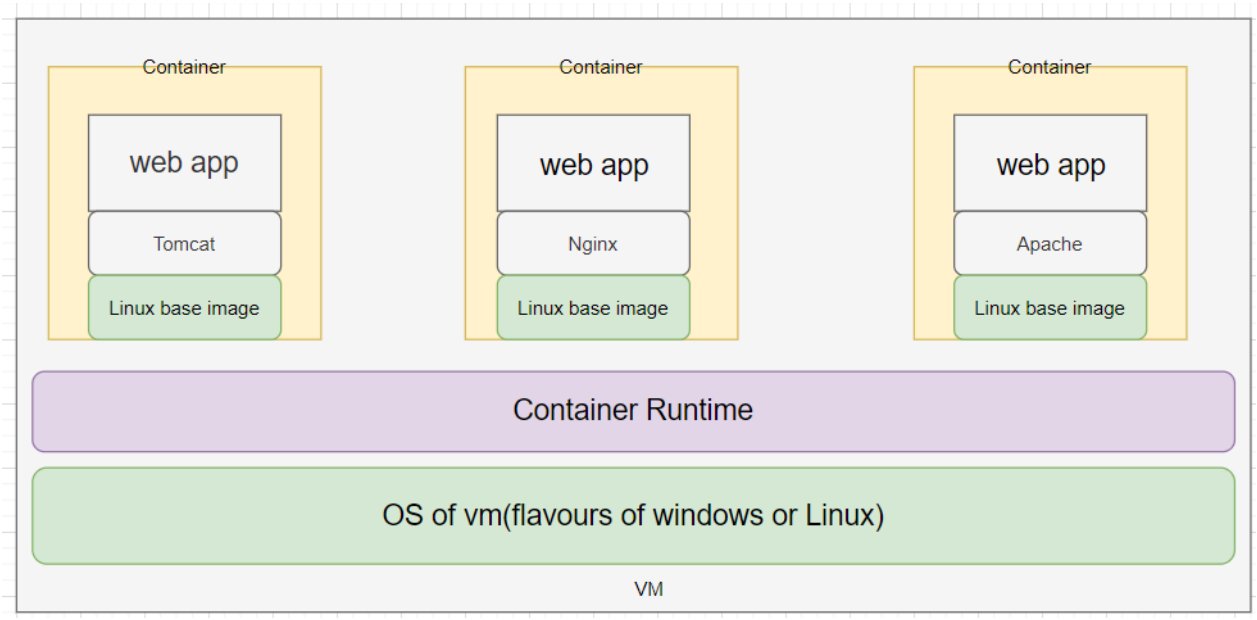

Figure 1: Docker Container Architecture

B. *CONTAINERS:*

Application and all its dependencies put together are known as a container. Multiple containers can be mad to run on a single VM as long as they belong to the same kernel either Windows/Linux. Containers can be built as a windows container or a Linux container. And applications will behave the exact same way in all environments. It is also very easy to maintain micro services along with their configurations [2].

III. KUBERNETES

Kubernetes is a system for running and coordinating containerized application across a cluster of machines. K8S can be used to manage the lifecycle of containerized applications, scalability and also ensures high availability. We can define how our application should interact with other applications and with the outside world. We can scale applications up or down, perform rolling updates and switch traffic between different versions of the application to test features.

A. KUBERNETES ARCHITECTURE:

Kubernetes puts together individual physical or virtual machines into a cluster using a shared network to communicate with each other. All kubernets components and workloads are configured in the cluster.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

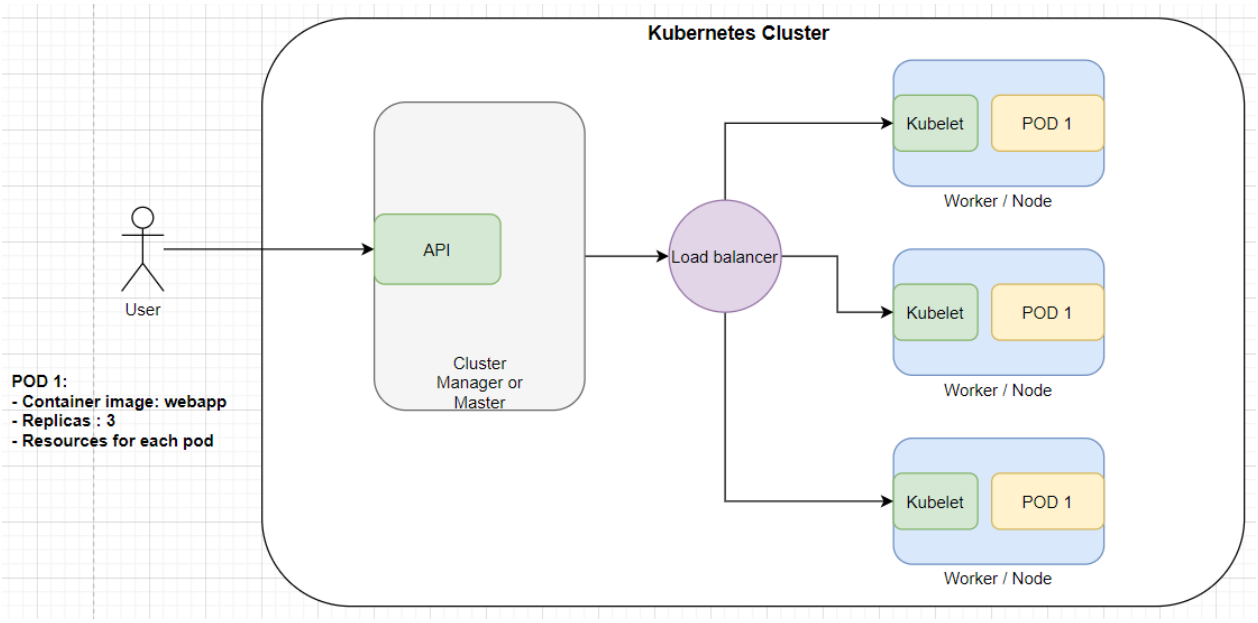

Figure 2: Kubernetes Architecture

There are two major components of a kubernetes cluster, the master and the nodes. The **master** is like the brain of the k8s cluster it exposes API for the clients, checking the health of the nodes, splitting up assigned tasks and managing communication between components[3]. It takes care of handling deployments and managing worker nodes. Azure provides cluster manager for free of cost.

Nodes are machines in which the work loads are run i.e. nodes are VMs in which actual deployment happens. Applications are run as containers, so each node must have a container runtime. Kubelet is a service that is running in the nodes that is responsible for taking up tasks from the master and creates or destroys containers accordingly. K8s is responsible for maintaining desired state configuration.

To deploy an application YAML files are used. These files contain information like what to create and how it should be managed in other words this file is the desired state configuration. The master node takes up this plan and decides how to run it in the existing infrastructure.

B. *Master Server Components:*

1. Etcd: etcd is used to store configuration data that can be accessed by all nodes of the cluster. It helps in maintaining cluster state. It helps services to configure according to up-to-date information.
2. Kube api server: It helps to assign workloads to nodes analyzing the current infrastructure and places work on acceptable node.
3. Cloud controller manger: K8s can be deployed in many different environments and can interact with various infrastructure providers to manage state of resources. Cloud controller manager acts as a bridge that allows K8s to interact with various providers with different features and APIs, this allows K8s to update its state information according to information gathered from the cloud provider.

C. *Node server components:*

1. Container runtime: Container Runtime is a mandatory component in every node that is made available when docker is installed. It helps managing the containers.
2. Kubelet: Kubelet is a service that runs in all nodes that acts as a contact point for the master. Kubelet receives commands from the master and begins the work, it also interacts with etcd to maintain states. Kubelet process takes responsibility for maintaining state of work in the node. It controls the container runtime to launch and destroy containers as needed.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

3. Kube-proxy: Kube-proxy navigates requests to the correct containers in the node it can perform a basic level of load balancing. It makes sure that services are available to other components.

D. *Kubernetes Objects and workloads:*

Kubernetes uses additional layers of abstraction over containers to provide features like scaling, life cycle management.

1. Pods: A pod is the most basic unit of kubernetes. A pod is generally one or more tightly coupled containers that are to be deployed in the same node. Pods are fundamental units of deployment. Container instances are not deployed directly, when pods are deployed container instances are a part of that pod.
2. Replication set and replication controller: These are used to horizontally scale applications. Replication defines a pod template and parameters to deploy identical pods. It can be used to either increase or decrease the number of running nodes. Controller is responsible for maintaining the specified number of pods or instances running. If one instance fails controller automatically start new pod to meet the desired state configuration.
3. Service: A service groups logical collections of pods to represent them as a single entity. This allows us to deploy a service that can monitor all containers of a particular type. IP address remains the same regardless of changes pods it route to. In order to give access to a pod to other applications or the end user a service has to be configured. the **Load Balancer** service type creates an external load balancer to route to the service using a cloud provider's Kubernetes load balancer integration. The cloud controller manager will create the appropriate resource and configure it using the internal service service addresses.
4. Deployments: Deployment is the easiest and most used resource for deploying your application. It is a Kubernetes controller that matches the current state of your cluster to the desired state mentioned in the Deployment manifest. e.g. If you create a deployment with 1 replica, it will check that the desired state of ReplicaSet is 1 and current state is 0, so it will create a ReplicaSet, which will further create the pod. If you create a deployment with name **counter**, it will create a ReplicaSet with name **counter-<replica-set-id>**, which will further create a Pod with name **counter-<replica-set->-<pod-id>.** Deployments are usually used for stateless applications. However, you can save the state of deployment by attaching a Persistent Volume to it and make it stateful, but all the pods of a deployment will be sharing the same Volume and data across all of them will be same.

IV.AZURE KUBERNETES SERVICE

Azure kubernetes service paves a simple way to deploy and manage a kubernetes cluster. AKS abstracts all the complexities and operational overheads of maintaining a kubernets cluster as azure takes care of most of the responsibilities [4]. As a hosted Kubernetes service, Azure handles critical tasks like health monitoring and maintenance for you. The Kubernetes masters are managed by Azure. You only manage and maintain the agent nodes. As a managed Kubernetes service, AKS is free - you only pay for the agent nodes within your clusters, not for the masters. You can create an AKS cluster in the Azure portal, with the Azure CLI, or template driven deployment options such as Resource Manager templates and Terraform. When you deploy an AKS cluster, the Kubernetes master and all nodes are deployed and configured for you. Additional features such as advanced networking, Azure Active Directory integration, and monitoring can also be configured during the deployment process. Windows Server containers support is currently in preview in AKS [5].

A. *Features of aks:*

- *Identity and security management***:**

To limit access to cluster resources, AKS supports Kubernetes role-based access control (RBAC). RBAC lets you control access to Kubernetes resources and namespaces, and permissions to those resources. You can also configure an AKS cluster to integrate with Azure Active Directory (AD). With Azure AD integration, Kubernetes access can be configured based on existing identity and group membership. Your existing Azure AD users and groups can be provided access to AKS resources and with an integrated sign-on experience.

- *Integrated logging and monitoring:*

To understand how your AKS cluster and deployed applications are performing, Azure Monitor for container health collects memory and processor metrics from containers, nodes, and controllers. Container logs are

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

available, and you can also review the Kubernetes master logs. This monitoring data is stored in an Azure Log Analytics workspace, and is available through the Azure portal, Azure CLI, or a REST endpoint.

- *Auto scaling:*

To keep up with application demands in Azure Kubernetes Service (AKS), you may need to adjust the number of nodes that run your workloads. The cluster autoscaler component can watch for pods in your cluster that can't be scheduled because of resource constraints. When issues are detected, the number of nodes in a node pool is increased to meet the application demand. Nodes are also regularly checked for a lack of running pods, with the number of nodes then decreased as needed. This ability to automatically scale up or down the number of nodes in your AKS cluster lets you run an efficient, cost-effective cluster.

V. SETTING UP THE CLUSTER

A. Terraform:

Terraform is a tool for building, changing, and versioning infrastructure safely and efficiently. Terraform can manage existing and popular service providers as well as custom in- house solutions. Configuration files describe to Terraform the components needed to run a single application or your entire datacenter [6]. Terraform generates an execution plan describing what it will do to reach the desired state, and then executes it to build the described infrastructure. As the configuration changes, terraform is able to determine what changed and create incremental execution plans which can be applied. The infrastructure Terraform can manage includes low-level components such as compute instances, storage, and networking, as well as high-level components such as DNS entries, SaaS features, etc.

Terraform can be run from your local machine or from azure cloud shell. In either ways it works the same. When creating any resource on the cloud using terraform takes you a step further ahead. Before actually deploying terraform plan gives you an over view of what changes will we made if the script is applied. Terraform also helps create multiple resources using a single terraform file. We don’t have to worry about what order the resources should be specified terraform will automatically detect the dependencies and start deploying the resources either parallel or in a sequential fashion.

B. *Accessing the cluster:*

Once the kubernetes cluster is up and running. The cluster can be accessed via kubectl. Kubectl is kubernetes cli version which can be used to get information about the cluster and all the workloads on the cluster it can also be used to make deployments to the cluster using yaml files. The application in the cluster can be exposed by creating a load balancer and exposing its external IP address.

VI. RESULTS AND DISCUSSION

The objective of this project was to deploy a dynamic webpage in kubernetes which has been successfully achieved. The application runs as expected in the cluster and is available to access from anywhere and the traffic flows to the pods in a well-balanced manner. The application can scale up or down and rolling updates can be performed manually. Kubernetes is a vast subject and there is always more to explore and work in it. In future this project can be further updated in several ways, one of which is using Jenkins X to automate CI/CD pipeline instead of shell scripts. Jenkins X can be integrated in the project in such a way that when any change is made to the application and when the developer commits these changes to github it automatically gets reflected in the cluster. Further several name spaces may be created in the cluster for the purpose of testing and resource quota can be fixed for each namespace. Volumes can be attached to make the application work much faster and make it more reliable. We have only used AKS for this project but all cloud providers including AWS and Google Cloud platform provide Kubernetes service we can look into all the additional features these cloud providers offer and choose the best one to create our cluster in. Kubernetes being a latest technology always has room for new updates.

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

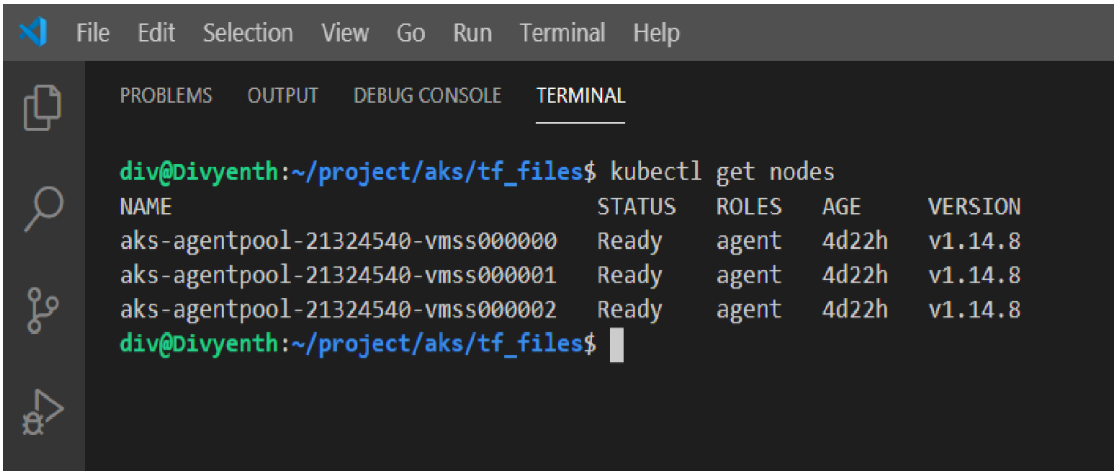

Figure 3: Node Status

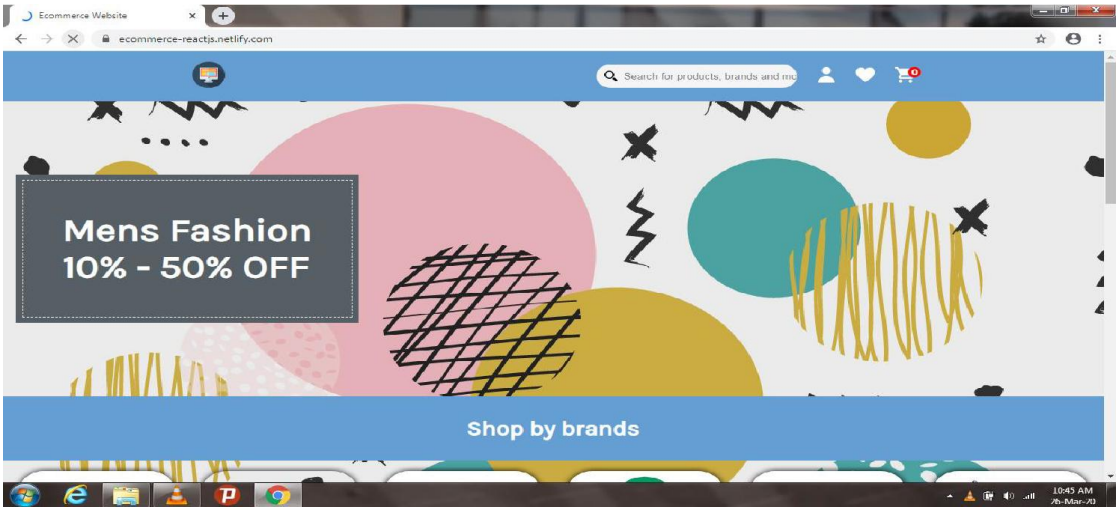

Figure 4: Landing Page

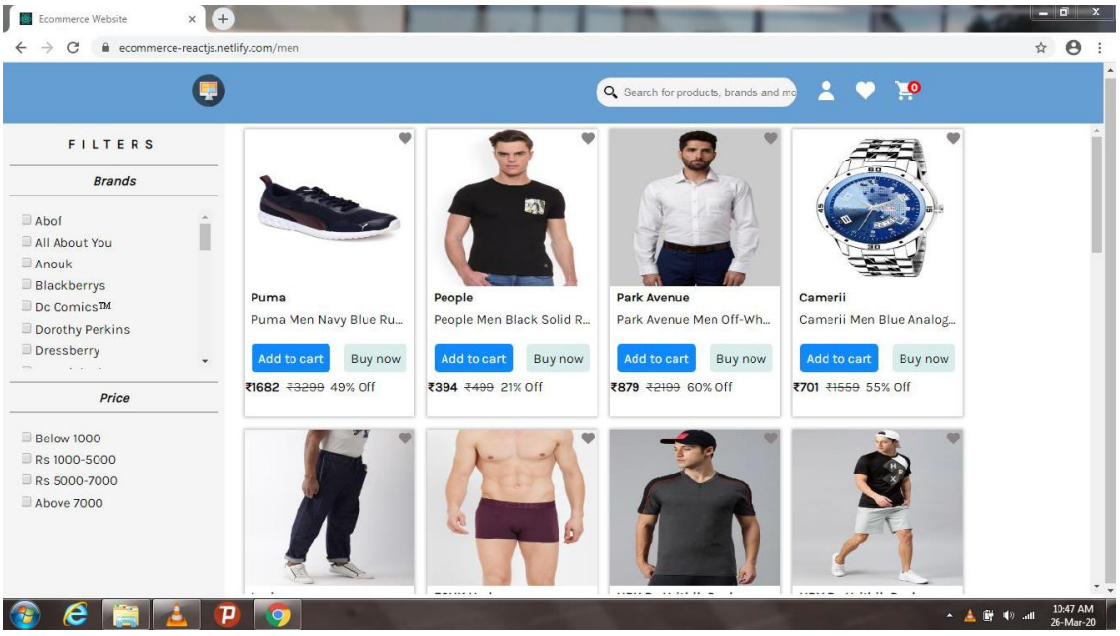

Figure 5: Men Products

| e-ISSN: 2320-9801, p-ISSN: 2320-9798| www.ijircce.com | Impact Factor: 7.194 |

||Volume 8, Issue 4, April 2020||

VII.CONCLUSION AND FUTURE WORK

Application development and deployment has seen a lot of changes in the reset past. With more and more companies adopting to cloud technologies it is important to cope up with these changes. Kubernetes and docker have entirely changed the word of application development and deployment and are able to satisfy the needs of the companies, providing loads of features for ease of work. Tools like Jenkins and Jenkins x help automate all the deployment activities in the CI/CD pipeline. Deploying application on the cloud has definitely helped overcome all the pitfalls and overheads of the traditional techniques. For further enhancement of the project azure autoscaler and monitoring tolls can be looked into for better understanding and working of the cluster.

REFERENCES

1. T. Harter et al., "Slacker: Fast distribution with lazy Docker containers", *Proc. Usenix FAST*, 2016.
2. S. Hoque et al., "Towards container orchestration in fog computing infrastructures", *Proc. IEEE COMPSAC*, 2017.
3. V Singh, S K. Peddoju, "Container-based microservice architecture for cloud applications[C]", *Computing Communication and Automation (ICCCA) 2017 International Conference*, pp. 847-852, 2017.
4. Azure kubernetes service (aks). highly available secure and fully managed kubernetes service, June 2019
5. Y. Pan, I. Chen, F. Brasileiro, G. Jayaputera, R. O. Sinnott, "Performance and overhead study of containers running on top of virtual machines", *A Performance Comparison of Cloud-based Container Orchestration Tools*, June 2019.
6. P. Di Tommaso, M. Chatzou, E.W. Floden, P.P. Barja, E. Palumbo, C. Notredame, "Nextflow enables reproducible computational workflows", *Nature Biotechnology*, vol. 35, pp. 316, 2017.