Multi Cloud Deploymentwith Containers

Baskaran Jambunathan[#1], Dr.Y.Kalpana[#2]

[#1]Research Scholar,
Dept. of Computing Science,
Vels University, Chennai,Tamil Nadu, India.
[#2] Associate Professor
Dept. of Computing Science,
Vels University, Chennai,Tamilnadu, India.
[1] shriyabaskaran@yahoo.com
[2]ykalpanaravi@gmail.com

***Abstract:*Micro services are one of the essential aspects of Cloud native application development in distributed environment. Industries are looking for more agile, innovative and rapid application development to meet their business needs. In addition, these applications needs to be deployed in a distributed multi cloud environment and has to address the essentials of Scalability, Portability and security. Also people need to understand how to convert a monolithic to micro services architecture in building real time applications. We would like to address in this paper on how to design such applications, and portability of such applications using containers across multi cloud environment and its challenges.**

Key Words: Micro Services, Monolithic, Dockers, Distribution, Kubernetes, Multi cloud

I. INTRODUCTION

Current trend in Industry is thatapplications need to be more agile in nature, flexible and adaptive to changes. In addition, modern Application development and deployment needs to be distributed, scalable and portable across different platforms seamlessly to meet the business demands. Applications developed in the past, are in general, monolithic in nature and are rigid and tightly coupled. Any small changes to be made in any component in this application will have challenges and have large impact on productivity, time, cost and deployment. Since business demands these applications are to be distributed, scalable and portable across different technologies and platforms, it is more complicated for developers to make changes, test and deploy these applications in short turnaround time.

Micro services Architecture address these problems to large extend. The essence of Micro services is to decompose the application components in to small, logically grouped functionality as services,each running in its own process and communicating with lightweight mechanisms, often an HTTP resource API. These services are built around business capabilities and independently deployable by fully automated deployment mechanism. There is a bare minimum of centralized management of these services, which may be written in different programming languages and use different data storage technologies.

Industries are moving their applications today to different cloud platforms to improve their business performance and reduce cost. Applications are migrated to either public or private or to a Hybrid model. Depending on the technology used and business needs, different cloud platform is selected and their applications are migrated to the selected platform either AS-IS or undergoing changes. Current business demands the application has to be flexible with technologies and to be ported across different cloud platforms seamlessly with less effort and changes. Micro services architecture address most of these problems and provide facilities to make changes without impacting other components and with support of Docker containers, it is easy to port these applications across platform as they are lightweight, self-contained components. We will discuss on how to design micro services, its challenges and how containers helping them to be portable across multi cloud environment.

II. MIGRATION TO PAAS ENVIRONMENT

Platform as a Service (PaaS) [1] is one of the cloud services used for hosting and building application on the desired platform. Here the features of platform are extended and deployed as services and the applications hosted on this platform can consume these services. The services are made available as APIs and webservices which provide interfaces for application, database development, storage, and testing. This allows businesses to streamline the development, maintenance and support of custom applications, lowering IT costs and minimizing the need for hardware, software, and hosting environments.

There are set of services which the Public and private cloud Service providers are providing for any consumer to use and meter it as per usage. But most of the enterprise has huge number of platform and infrastructure and made lots of investment. In addition they have developed large number of applications and data which are very sensitive to be migrated on to public cloud and hence leading to have their own private cloud platform.

Enterprises which have more sensitive data and mission critical applications which cannot be migrated to Public cloud are looking to have their own private cloud to be set up and deploy their application over private cloud. Few Enterprise are even looking to have a hybrid cloud [1] model to have part of the application in public and part in private cloud and make use of best of both cloud delivery models.

A. Enterprise PaaS Platform

Large number of enterprise today, because of security and operational challenges, are critical on moving to public cloud and wanted to have their own Enterprise cloud (PrivateCloud) setup using the resources in their datacenter. They wanted to have their own enterprise PaaS [1] platform on top of IaaS [1] layer created using their own infrastructure.

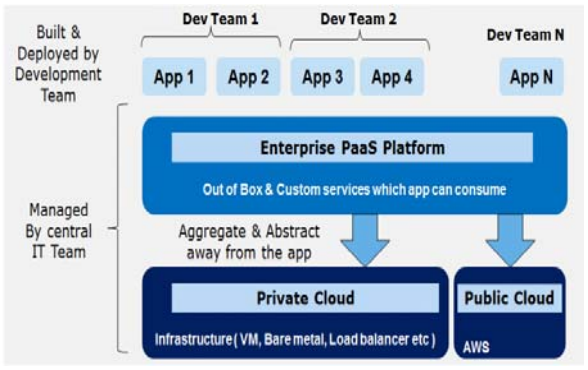
Built & Deployed by Development Team
Dev Team 1
Dev Team 2
Dev Team N
App 1
App 2
App 3
App 4
App N
Enterprise PaaS Platform
Out of Box & Custom services which app can consume
Managed By central IT Team
Aggregate & Abstract away from the app
Private Cloud
Infrastructure (VM, Bare metal, Load balancer etc)
Public Cloud
AWS

Fig.1. Enterprise PaaS Platform

Following are the key reasons for Enterprise PaaS

- Facilitates development, deployment and operations for IT on a private infrastructure.
- Provides greater security, compliance and governance controls for developers.
- Isolate developers from Operational tasks through platform automation.

In addition, Enterprise PaaS has lots of business benefits to companies like Increased Agility, Reduced Cost, Reduced Complexity, Streamlined Application Management, Cloud agnostic – portability and Horizontal scalability. Enterprise PaaS platforms are hosted on IaaS layer and are abstracted from the under laying infrastructure. These platform has containers which are secured, independent and scalable. Applications which are developed as micro services are deployed in this containers. We wanted to elaborate how micro services and containers are helping in cloud agnostic portability and how they are effectively orchestrated with Dockers [2] and Kubernetes [2].

III. MICRO SERVICES – QUICK OVERVIEW

In past decades enterprise application developers are used to develop large monolithic applications which consist of large number of components and services as a composite, single large application. The key challenge is that the applications are undergoing changes due to change in business functionality and developers need to make changes, rebuild and re-deploy the whole application for even making a small change. Secondly, modern day environments are heterogeneous and distributed in nature and deploying and managing such composite large applications are more difficult in a distributed environment.

Micro services [2] address this problem to large extends. Micro services is a software development paradigm in which complex applications are composed into small, independent processes communicating with each other using language-agnostic APIs. These services are built as separate components and are organized around capabilities so that they are easy to replace. Micro services can be implemented using different programming languages, databases depending on what is best for the given functionality and business needs. It lends itself into continuous delivery process and are distinct from Service oriented architecture (SOA) [2].

*A. Monolithic Applications – Issues and Challenge*s

Traditionally over the last few decades, enterprise develop monolithic applications with Service oriented architecture (SoA) to help application to improve scalability, availability, reliability and performance. These applications if it has to undergo changes, it has to be re-built again after changes and has to be re-deployed as a whole again. This has major challenges in terms of impacting developer productivity as it takes more time to understand the code as a whole to make changes. Continuous integration [2] and continuous deployment [2][3] is difficult because in order to deploy one component, we have to deploy the whole application again.

Other Key challenges are

a) Difficult to make change in the technology stack of the application as we are struck with one technology
b) Scaling the application is difficult at times as it can scale one dimensionally.

c) As the data grows, caching is difficult and increase memory consumption
d) Applications are centralized and owned by an organization or an individual and leads to lots of dependency.

B. Micro services – Its Characteristics

Micro services on the other hand decomposes the application into set of collaborating services with each service implement a set of narrowly related functions. These services communicate with each other with both synchronous and asynchronous protocol.

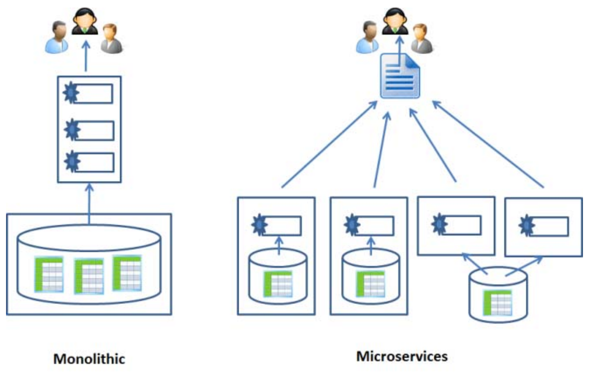

Fig 2. Monolithic and Micro services

Services are developed and deployed independently and each service has its own database in order to be decoupled. Moreover component (Components is a unit of software that are independently replaceable and upgradable) are deployed independently and are light weight in nature. Micro service architectures will use libraries, but their primary way of componentizing their own software is by breaking down into services. We define libraries as components that are linked into a program and called using in-memory function calls, while services are out-of-process components who communicate with a mechanism such as a web service request, or remote procedure call

Following are the Benefits of micro services

a) Each Microservice is small, easy to develop and deploy and can be deployed independently.
b) Easy to scale development
c) Improved fault isolation – if there is a memory leak in one service, only that service is isolated
d) Eliminate long term commitment to technical stack.

Key Drawbacks

a) Developers has to deal with additional complexity of dealing with distributed system
b) Testing the application is more difficult
c) Developer must implement interservice communication interfaces and any change in interfaces will impact others as they are inter-related.
d) If we have to implement use case which span across multiple services then managing distributed transaction is difficult and need to carefully co-ordinate between teams
e) Deploy and management of different types of services becomes difficult when the number of services increases.
f) Increased memory consumption with N application instances to interact with M services leads to N X M complexity
g) Increases Operational complexity and requires high level of automation and needs DevOps[3] skills.

IV. MICROSERVICE AND CONTAINERS

Containers [2] are usually described as light-weight runtime environments with many of the core components of a virtual machine and isolated services of an operating system designed to make packaging easy and execute these micro-services smoothly. Distributed application deployment demands multiple environments for the application components to be deployed and executed. These components need to be managed by a master and the request has to be distributed with load balancer[2]. These components are to be scaled as the components or services increases and needs to be secured in communication between environments. Containers address these two problems effectively. The Linux containers[2][3] are highly secured and the platform managing the containers helps them in auto-scaling and request processing. Applications developed in micro services are deployed with each service or a group of services deployed in each container and with more services, more containers are added automatically by the hosting platform. These containers are light weight in nature and share the underlying Operating systems.

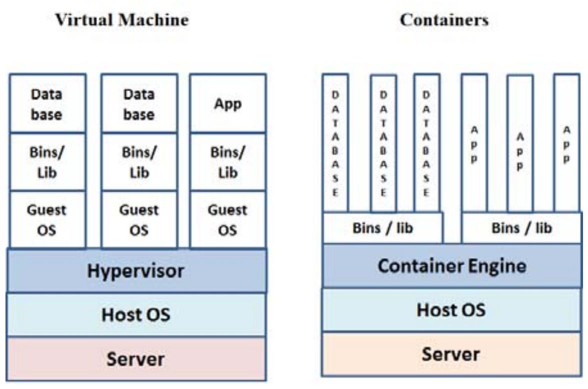

Fig 3. Virtual Machine and Containers

In Modern world, Companies are forced to make changes in their application rapidly to meet thebusiness demand and be competitive to sustain in business. Application development has to undergo lots of changes more frequently and needs to be managed and deployed on to multiple environments and has to support any cloud services. In addition, applications needs to adhere to the demand in security needs and are available to customers with high availability [3] and scalability which will have high user experience. Hence these applications are need to be portable across platform without much challenges and deployment process should be easier irrespective of which platform or cloud services application uses.

V. PORTABILITY USING DOCKER

Docker aims to automate the deployment of applications inside portable containers that are independent of hardware, host operating system, and language. In contrast with Virtual Machines, Docker containers do not include a guest operating system but share the operating system with other containers.

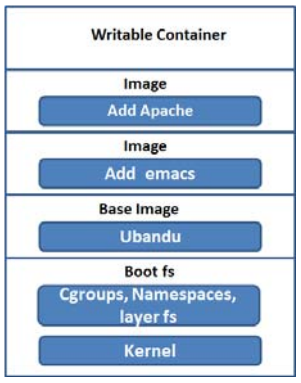

Fig 4.Docker Container

Docker provides features to do package an application and all of its dependencies in a virtual container that can run on any Linux server which Docker runs. Dockers are more secured, portable across platforms. There are many Private PaaS products available in the market are changing their internal architecture to adhere to Docker standard and helping application developers to develop in any language and package their application components with a mechanism which help them to deploy on to any cloud.

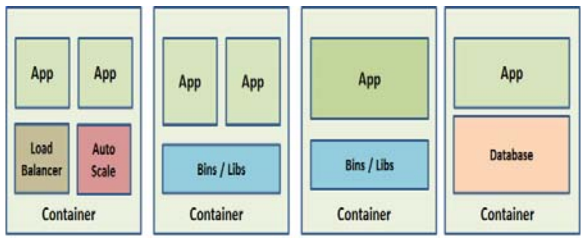

Fig 5: Container based Application Architecture

The power of Docker is an application-centric packaging model and the flexibility of an image-based deployment method, combined with the large and rapidly growing selection of images available in the Docker Hub [3] registry. This gives developers the broadest selection of components to create their applications and also enables portability across bare metal systems, virtual machines and private and public clouds. Docker packaging model enable users to leverage any application component packaged as a Dockerimage[3]. This will help developers to tap into the Docker Hub community to both access and share container images to use in any supported products.

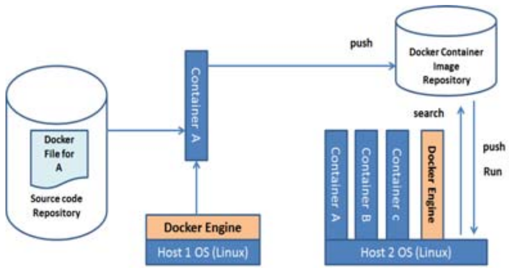

Fig 6 Deployment through Docker container

Dockers are lightweight, portable and self-sufficient so that application developers can build, package the application independently and deploy them in a distributed, multi-environment without any issues. This makes the application deployment easier and faster and helps organization to reduce cost by avoiding repetition and redundant operational tasks.

VI. MULTI CLOUD AND MICRO SERVICES

With high demand in business needs, organizations are forced to develop new applications, as part of innovations, to meet their business objectives to be in race with the competition and need to deploy the application on multiple platforms and devices to satisfy customer needs. Application deployment has to happen quite often and their time to market should be very less. Organization needs to standardize and optimize the deployment model which can be adopted all over, and can be streamlined so that everyone in the organization can follow the same without much difficulty in their journey of enterprise application management.Hence, organizations are looking for standardized deployment models as they have multiple environment to manage and application developed in different languages are to be portable across different platform (sometimes different data centers and cloud platforms), automatically and can be deployed with less effort.

Docker containers allow users to deploy applications in any environment (e.g., cloud, cluster, virtualized, bare metal, etc.) much faster and more efficient than virtual machines. As more companies look to adopt this technology, a pressing need to run these containers across multiple distributed infrastructures arises. In addition, relying on a single provider can lead to “provider lock-in” and restrict users from taking advantage of a competitive pricing market. The current efforts are focused on deploying containers at a single site/cloud level but companies believe that deploying containers across federated infrastructures is the future. When Containers carrying application components services deployed across different cloud needs to be orchestrated and managed effectively to address the distribution application deployments.

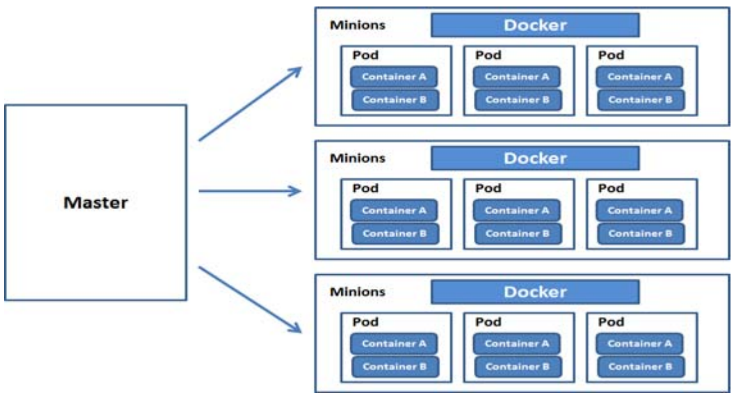

Fig 7.Kubernetes Orchestration mechanism

Kubernetes [3] is for managing containerized applications across multiple hosts, providing basic mechanisms for deployment, maintenance, and scaling of applications. It is a tool for container orchestration and provides a high level abstraction for networked applications on top of containers that allows easy scaling, service discovery, and deployment. Kubernetes manage the lifecycle like start, stop, update, and manage a cluster of machines running containers in a consistent and maintainable way. It is particularly suited for horizontally saleable, stateless, or 'micro services' application architectures and it does not mean others will not work or are ignored. It also has additional functionality to make containers easier to use in a cluster (reach ability and discovery).Clusters are compute resources on top of which our containers are built.

Pods are a collocated group of Docker containers with shared volumes. They're the smallest deployable units that can be created, scheduled, and managed with Kubernetes. Replication controllers manage the lifecycle of pods. They ensure that a specified number of pods are running at any given time, by creating or killing pods as required. Services provide a single, stable name and address for a set of pods. They act as basic load balancers. Labels are used to organize and select groups of objects based on key: value pairs. Kubernetes has a   Master and aset of kubelets [4] to manage containers and establish application connectivity between containers. It provides a high level abstraction for networked applications on top of containers that allows easy scaling, service discovery, and deployment.

The power of Docker is an application-centric packaging model and the flexibility of an image-based deployment method, combined with the large and rapidly growing selection of images available in the Docker Hub registry. This gives developers the broadest selection of components to create their applications and also enables portability across bare metal systems, virtual machines and private and public clouds.With Dockers packaging one can do to achieve multi-version, deployments and application isolation.

VII. MULTICLOUD DEPLOYMENT

Applications developed in micro services architectures has to be deployed in a distributed environment to host each of the services independently and has to have a communication mechanism to take care of inter service interactions and ensure the application functions holistically. In this scenario, the transaction, security and performance aspects have to be handled and managed efficiently as well..Containers provides the ideal platform for hosting these services and the platform managing these containers (PaaS Platform like Cloud Foundry [5]) are handling the security, portability, transactions and performance and in addition failover and application resilience. Docker containers which are light weight in nature can host different runtime environment and can have application built in different technologies and still can communicate with other containers effectively.

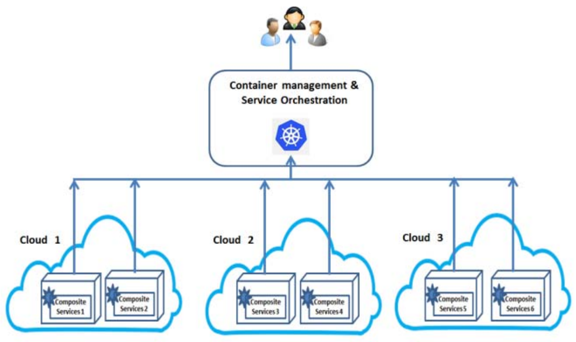

Fig 8. Multi Cloud Deployment

In order to manage these containers, we need an effective cluster management and Orchestration mechanism to control and manage these containers.Kubernetes [3]is one such techniquesdoes this job for us. Kubernetes is a tool to simplify containerized application deployment. By utilizing a policy framework, kubernetes provides the ability to scale on demand, and across multi-cloud clusters. This is enabled through container engines and management orchestration.

Multi cloud deployment is going to be the future for companies as it deployments allow users to take advantage of differences in various clouds, including price, performance, capability, and availability differences.As clouds with a higher-preference become available, the environment should balance, automatically downscaling, that is, terminating instances, in lower-preferred clouds, and up-scaling[3], launching instances in higher-preferred clouds.

Upscaling is typically automated by auto-scaling services like amazons auto scaling services, which allow users to define the number of instances that should be deployed in each cloud. However, downscaling [3] isn't typically automated and presents a number of unique challenges that must be addressed. For instance, users must be able to clearly define their preferences for different clouds and policies must be created. These policies should identify which clouds to terminate instances in, which instances to terminate, and how the instances should be terminated. Rebalancing policies should balance user requirements as well as workload and environment characteristics to avoid introducing excessive workload overhead. How to do rebalancing, up-scaling and down scaling are beyond scope of this paper. We focus more on how micro services and containers can help in multi cloud deployment and how kubernetes can manages these containers by means of orchestrations.

Containerization facilitates the step from a single host to clusters of container hosts to run containerized applications over multiple clusters in multiple clouds. The built-in interoperability makes this possible. Container-based cluster architecture groups hosts into clusters [2][5]. Fig. 7 .that illustrates an abstract architectural scenario based on common container and cluster concepts. Container hosts are linked into a cluster configuration. Cluster management has to address the following key aspects

a. Deployment of distributed application through containers
b. API for creating and managing lifecycle of the services in clusters
c. Service manager to look after software packing and management
d. Agent manages containers lifecycle

A cluster architecture is composed of engines to share service discovery (e.g., through shared distributed key value stores) and orchestration/deployment (load balancing, monitoring, scaling, and also file storage, deployment, pushing, pulling).Lightweightvirtualized cluster architecture should provide a number of management features as part of the abstraction on top of the container hosts. Some of them are like

a) Hosting containerized services and providing securecommunication between these services,
b) Auto-scalability and load balancing support,
c) Distributed and scalable service discovery andorchestration
d) Transfer/migration of service deployments betweenclusters.

Kubernetes is based on processes that run on Docker hosts that bind hosts into clusters and manage containers. Minions(slaves in Kubernetes clusters) are container hosts that run pods, i.e., sets of containers on the same host. Openshift [5] has adopted Kubernetes in their 3.0 release. Expertise by Google incorporated in Kubernetes competes here with platform-specific evolution towards container-based orchestration. Cloud Foundry [5], for instance, uses Diego[5] as a new orchestration engine for containers.

Network and data challenges are another set of problem in container management. In Kubernetes, each group of containers (called pods) receives itsown unique IP address, reachable from any other pod inthe cluster, whether co-located on the same physical machineor not. This requires advanced routing features based on network virtualization. To address data storage,what is needed is to pair up a container with a storage volume that, regardless of the container location in the cluster, follows it to the physical machine.

Industries in future would develop packaged application based on services, deploy them in multi cloud and across data centers with public, private and hybrid cloud using different services Infrastructure as a service (IaaS) providers and also will use multi-PaaS environment hosted on different IaaS layers. Multi cloud deployment is going to be inevitable for companies and in future all are compelled to move into this model. The envisioned multi cloud platform has self-service, cloud management, deployment — all built, and these applications enable enterprises to leverage cloud resources across platform successfully, securely, and profitably.

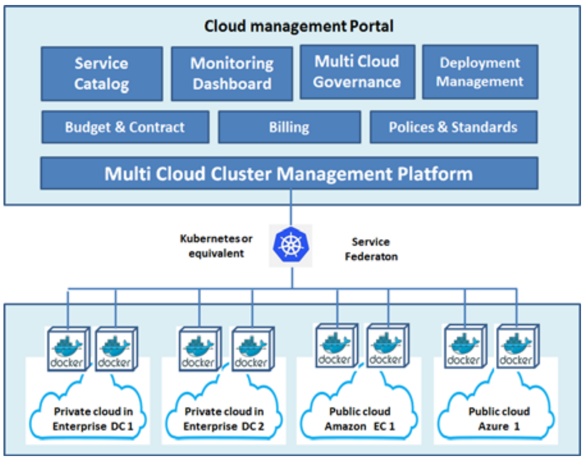

Fig 9. Multi Cloud management platform

By leveraging these abstractions, organizations avoid lock-in to a particular cloud provider and maintain portability to other cloud providers as they offer comparable capabilities. In addition, powerful multi-cloud governance module gives organization, visibility and control across private and public clouds.Research is on to address a deployment process model and standards for porting and deploying application across different clouds and makes the best use of the entire platform on both IaaS and PaaS layer.

VIII. FUTURE WORK

Cloud management platforms are still at an earlier stage than the container platforms that they build on. While clusters in general are about distribution, the question emerges as to which extent this distribution reaches the edge of the cloud. Container technology has a huge potential to substantially advance PaaS technology towards distributed heterogeneous clouds through light weightiness and interoperability. However, significant improvements are still required to deal with data and network management aspects as is providing an abstract development and architecture layer. Container cluster-based multi-PaaS is a solution for managing distributed software applications in the cloud, but this technology still faces challenges. While Docker has started to develop its own orchestration solution and Kubernetes is another relevant project, a more comprehensive solution that would tackle orchestration of complex application stacks could involve Docker orchestration based on the topology-based service orchestration standard TOSCA (Topology and Orchestration Specification for Cloud Applications [5]), which is for instance supported by the Cloudify PaaS.

Biarca [3][4], a silicon valley based company, is addressing a key need in the industry today, providing critical knowledge and expertise in Kubernetes to help enterprises design and deploy cloud native distributed applications across multiple clouds (private, public, hybrid) to improve agility and efficiency. Containers are a disruptive technology that will have a significant impact on IT in the coming years and tools like Kubernetes are critical in speeding up that transformation.

C-Ports [3][4] (pronounced seaports) has already been demonstrated to effectively deploy containers across different clouds in order to create a dynamic federation. Additionally, C-Ports is not tied to a specific container scheduler, i.e., it can work with any local container scheduler, such as Kubernetes or Bluemix [5], or directly deploy containers on the given resource/cloud, thereby increasing its portability and flexibility.

Our research is to explore this area further and arrive at the design process for container based orchestration and deploy application in multi-PaaS environment in multi-cloud environment. More to this, would like to explore the TOSCA standards, which is evolving, further. Key area to address in the multi cloud deployment is to have common topology description and orchestration, common interface to cloud service providers, process to address interoperability and portability and create a transformation model for deployment.

REFERENCES

[1] Rafael Moreno-Vozmediano, Ruben S. Montero, Ignacio M. Llorente. Multi-Cloud Deployment of Computing Clusters for Loosely-Coupled MTC Applications–Draft for EEE TPDS (special issue on many task computing), JULY 2010

[2] Antonio Brogi, Michela Fazzolari, Ahmad Ibrahim,Jacopo Soldani, PengWei Wang -Adaptive management of applications across multiple clouds: The SeaClouds Approach –CLEI Electronic journal, April 2015

[3] Jose Carrasco, Javier Cubo, and Ernesto Pimentel - Towards a flexible deployment of multi-cloud applications based on TOSCA and CAMP. University of Málaga, Dept. Lengujes and computer science, Spain

[4] Dmitry Duplyakin, Paul Marshall, Kate Keahey, Henry Tufo, Ali Alzabarah - Rebalancing in a Multi-Cloud Environment –University of Colorado, USA

[5] Claus Pahl -Containerization and the PaaS Cloud -Dublin City University, Dublin, Ireland

[6] Claus Pahl - Containers and Clusters for Edge CloudArchitectures – a Technology Review –Dublin City University, Dublin, Ireland