Orchestration vs PaaS vs CMP

Cloud Management Roundup

By: GigaSpaces Research, Cloudify Team
Contributors: Nati Shalom

Orchestration vs PaaS vs CMP

Table of Contents

Introduction

There are basically three main approaches to managing applications in a cloud environment: Orchestration, PaaS (Platform as a Service) and CMP (Cloud Management Platform). The end goal of all three is fairly similar - have a fully managed application on the cloud. It is therefore not surprising that users have a hard time choosing which of the approaches best suits their needs.

In this white paper, I wanted to make a simple distinction between these three options and offer some guidance on when you should consider each of the approaches.

Categories For Comparison

We will use the following categories as the basis for the comparison of the three different cloud management approaches:

Application Workloads

In this category we compare the three approaches based on the kind of application workload that best fits each platform, for example: web applications, stateful applications, big-data, legacy, etc.

Support for Advanced IaaS Services

In this category we compare the three approaches based on the depth of IaaS support, for example: how much each platform can utilize advances IaaS services such as DBaaS, LBaaS, EMR, etc.

Support for DevOps Processes

DevOps processes such as continuous deployment tend to be application specific as they touch not just the application itself, but also other services that are external to the application such as - support system, build-system etc. A typical DevOps process involves creating new environments for QA, Production, etc. or updating existing deployments. There are different techniques such as Canary, Blue/Green, or Immutable that are often used to handle those processes. In this category we compare the three approaches based on the degree of support for those DevOps patterns.

Support for Containers and Cloud Native Stack

Containers can be viewed as lightweight VMs or application packages. The use of containers removes a large part of the application configuration management and packaging complexity. In this category, we compare the three approaches based on the degree of support for that container technology e.g. Docker, Swarm, Kubernetes, Mesos, Fleet.

Support for Bare Metal

Bare Metal machines are often used to allow maximum performance and utilization of the HW resources by bypassing the virtualization layer overhead. Bare metal is becoming a popular target for application deployment with the introduction of new bare metal cloud as well as containers that provide a cost effective way to use bare metal resources. In this category we compare the three approaches based on their support for a bare metal environment.

Support for Network Orchestration

Many enterprises and clouds support virtual networking through SDN. In this category we compare the three approaches based on the degree of support for network orchestration.

Target Users

This category will compare the different approach fits well with Developers in a DevOps role as well as Operations.

Let's take a deep dive into the different cloud management options.

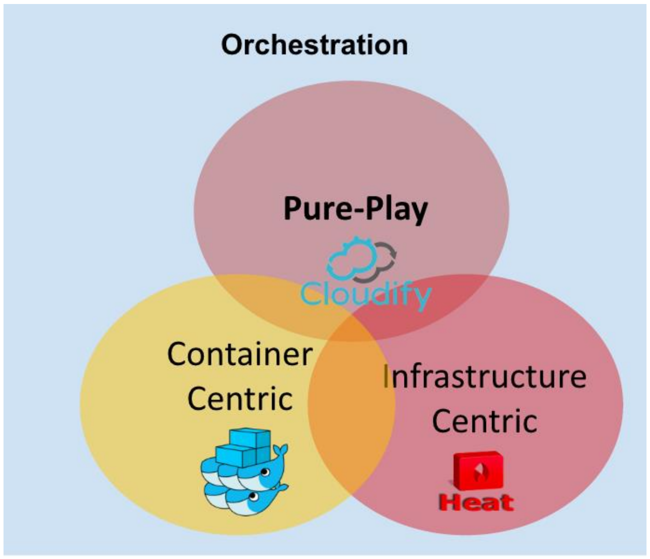
Orchestration Defined

Definition
Cloud Orchestration is basically a method for automating manual IT processes such as provisioning, installation, configuration management, remediation, maintenance, monitoring, scaling, etc.

- **Application Workloads**:
  The automation approach is fairly un-opinionated and therefore can address any use case that can be mapped into distinct manual steps. Because of this, it can be applied to a large variety of workloads starting from simple web applications to Big Data and analytics, or even legacy applications.

- **Support for Advanced IaaS Services**:
  Orchestration/automation doesn't limit you from the use of any IaaS services as it doesn't impose a layer of abstraction to manage a hybrid cloud environment.

- **Support for DevOps Processes**:
  Orchestration models the application into blueprints which makes it easy to create new instances of the same blueprint for Dev/Test or Production in a consistent way (a blueprint represents an environment meta-model). Orchestration also provides a way to interact with the existing deployments through workflows. Workflows are used to implement the DevOps processes. Since workflows are basically an execution script, they can easily interact with any external system such as build system or support system as part of the continuous deployment process.



- **Support for Containers and Cloud Native Stack**:
  Most of the container based platforms (Docker, CoreOS) come with built-in orchestration. There's also other types of container-centric orchestration such as Kubernetes which provides a more complete solution for managing applications.

  Other "pure play" orchestration tools such as Cloudify and Terraform provide support for setting up a container-based infrastructure and also integrate a hybrid stack of multiple containers technologies such as Kubernetes and Mesos, alongside non-container stack technologies such as databases, Big Data and legacy apps.

- **Support for Bare Metal**:
  Orchestrators are basically automation tools and can orchestrate applications on a cloud-based environment or bare metal. Some of the orchestration uses a resource pool mechanism to allow dynamic allocation of the application resources on a pool of bare metal machines.

- **Support for Network Orchestration**:
  Network orchestration is considered a core piece in NFV. There are two modeling languages that are commonly used for network orchestration: TOSCA, a standard for application topology definition, and YANG, a standard modeling language for configuration of network devices. The two modeling languages can also be combined to deploy network-centric services.

- **Target Users**:
  The automation approach fits well for Developers in a DevOps role as well as Operations engineers.

Additional Reference: Orchestration Tool Roundup - Docker Swarm vs. Kubernetes, TerraForm vs. TOSCA/Cloudify vs. Heat

PaaS Defined

Definition
PaaS takes a more a developer-centric approach. It was built as an abstraction layer that is aimed to help the developers focus on writing code by abstracting all the infrastructure and operational aspects. To achieve this goal, most of the PaaS platforms come with a fairly opinionated approach on how applications should be running on those platforms (Cloud Foundry refers to this as the 12 factors).

- **Application Workloads**:
  PaaS products are mostly suited for greenfield applications that fit the 12 factors definition. They are also suited mostly to web application and not so much to Big Data or legacy applications.

- **Support for Advance IaaS Services**:
  The abstraction approach provided by most PaaS products aims to "hide" the complexity of the infrastructure from the developers through a relatively "thick" layer of abstraction. One of the side effects that comes with that abstraction approach is it limits the use of the infrastructure to a basic compute and storage pool of resources where in reality many of the modern cloud infrastructures today provide much more advanced services such as DBaaS, LBaaS, EMR etc. This also causes lots of duplication of logic for handling things like user management, billing, and quota, that are already provided by the infrastructure.

  In addition, a PaaS often comes with its own set of logging, monitoring and development tools. DevOps users on the other hand tend to build their own tool chain, and the set of tools they use tend to vary between users, and over time, quite rapidly.

- **Support for DevOps Processes**:
  Most PaaS products provide built-in implementations of some of the continuous deployment processes, e.g. Canary, Blue/Green. However, those implementations tend to be tied to the platform with only a

fairly limited degree of customization as the user is not expected to have access to the internals of the platform such as the load-balancer.

- **Support for Containers and Cloud Native Stack**:
  Both OpenShift and Cloud Foundry are built on top of containers. OpenShift, specifically, is built on top of Kubernetes. Having said that, PaaS doesn't provide direct control over the underlying container orchestration and therefore limits the use of container orchestration within the PaaS platform.

  On top of that, PaaS rely on a specific container technology, therefore users don't have the flexibility to use their container orchestration of choice such as Core-OS Fleet, Docker Swarm, Mesos.

- **Support for Bare Metal**:
  Most of the PaaS solutions were designed to run on top of a virtualized cloud-based environment and were not built to run on bare metal.

- **Support for Network Orchestration**:
  PaaS doesn't expose most of the aspects of the network configuration to the end user and therefore comes with their own, opinionated, network configuration architecture.

- **Target Users**:
  Developers

Additional Reference: PaaS vs Orchestration - [PaaS vs Docker]

CMP Defined

Definition
CMP stands for Cloud Management Platform. CMPs take an infrastructure-centric approach where the main focus is monitoring and managing of infrastructure resources such as virtual machines, storage, network etc. It can be used indirectly to manage applications by combining some orchestration capabilities as part of the platform.

- **Application Workloads**:
  Similar to automation, CMPs come less opinionated and can therefore address a wide range of applications. Having said that, many CMPs were designed to provide infrastructure management and control the virtual machine that hosts the application, but not the application itself. CMPs don't come with strong application management and automation capabilities to handle certain aspects such as dependency management, discovery, configuration management, application management, etc.

Automating the full lifecycle of a given application, including configuration management and automation of post deployment aspects such as fail-over and scaling, would require more integration work with third party tools in order to fill this gap.

- **Support for Advanced IaaS Services**:
  The main value of Multi-Cloud CMPs is that they often provide a “single pane of glass” for controlling and monitoring cloud infrastructure across different cloud providers. To achieve this goal, many of the CMPs have to force some degree of “least common denominator” abstraction where they view the cloud infrastructure as a simple pool of compute and storage resources. By doing so, they limit the use of more advanced services provided by most of the modern cloud infrastructures today, as mentioned above.

- **Support for DevOps Processes**:
  Most of the CMP products focus on managing infrastructure resources. Continuous deployment processes tend to be application-centric by definition. Handling DevOps processes through a CMP is often non-trivial and requires integration with third party tools to handle this task, but quite often, the level of interaction needed between the two systems requires tight integration.

  In addition to that CMPs often come with a fairly monolithic architecture that includes their own Monitoring, Logging, Billing, etc and therefore doesn’t fit well in a DevOps environment. DevOps users tend to build their own tool chain and the set of tools that they would use tend to vary between users, and over time, quite rapidly.

- **Support for Containers and Cloud Native Stack**:
  Most CMP can run containers on top of VMs.

- **Support for Bare Metal**:
  Most of the CMP solutions were designed to manage virtualized, cloud-based environments and were not built to manage bare metal resources.

- **Support for Network Orchestration**:
  Most of the multi-cloud CMP solutions provide a limited set of network configuration mostly related to configuration of security groups and the load balancer. Advanced networking configuration, such as creating private network per apps, micro-segmentation, routers, firewall, and WAN gateways, are often not supported.

- **Target Users**:
  Operations

Additional Reference: DevOps is not a Feature

Application Delivery

The purpose of this comparison was to demonstrate the differences between the three approaches for managing applications on a cloud-based environment. All of these approaches are a means to an end, and one of the most common use cases for these management frameworks is to allow faster delivery of applications.

To allow faster delivery of applications we need to provide the developers with the right tools for the job. Every developer may have different needs and, therefore, tools needed to serve his application as I pointed out in one of my previous posts entitled "What Developers Want".

Based on the above analysis, we can measure each of the approaches under the following criteria:

The Key Benchmarks

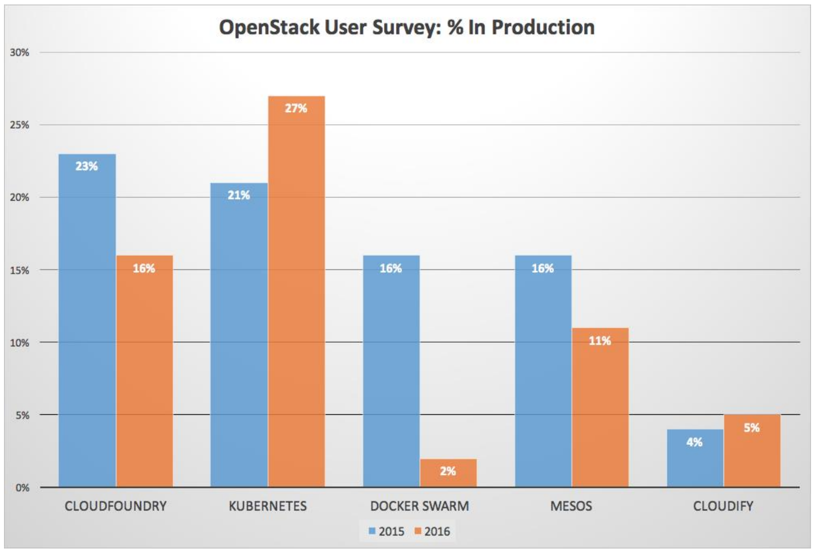
- **How fast you can serve the tools that developers need to become productive?**

The open-source world offers plenty of new frameworks and tools which are being announced almost on a daily basis. Developers want to have access to them all, as, in many cases, those tools will allow them to gain better productivity and thus speed up the development process. What developers care less about is the process of managing and configuring those tools.

- **Choosing a specific tool, such as a specific PaaS, is not a strategy.**

Don't build a strategy based on a specific tool, as by the time you are ready to support it, a new platform will emerge. Instead, be ready to support multiple platforms and tools even if they overlap to allow maximum flexibility for developers to then choose the right tool for the job rather that forcing them to fit into a specific platform stack.

As we can see in the following OpenStack surveys from 2015 and 2016, the popularity of the various platforms tends to change rapidly. In this specific case, we can see that Kubernetes grew almost 30% at the expense of Cloud Foundry. Cloudify grew 20% in use with OpenStack in production, according to users.

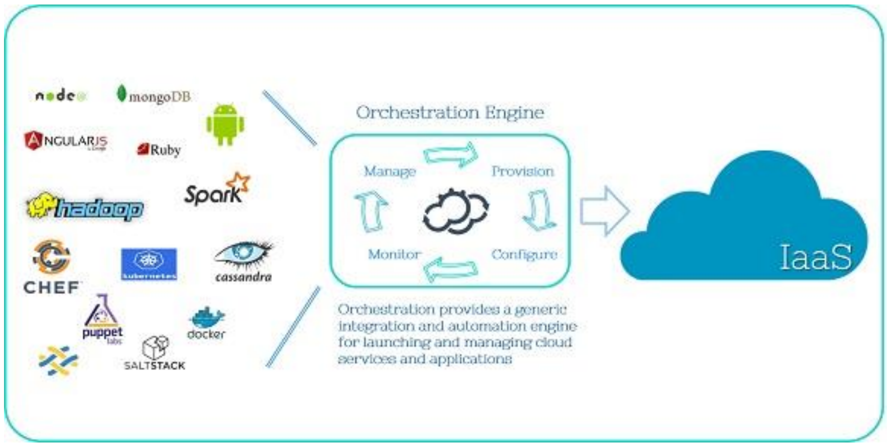


So, what we can understand from this analysis is that in order to serve those developers well the challenges are:

1. How fast you can introduce a new framework into your cloud?

2. How capable are you at managing many different types of frameworks?

It becomes clear now that in order to achieve this goal we need to have a generic way that will allow us to take any application and framework and offer it as a managed service.



Conclusion

In my personal opinion, the ability to introduce new frameworks fast is directly related to the flexibility of the management platform. Orchestration frameworks are more targeted to this approach because they are built to automate manual processes - in this case, installation, configuration, etc – directly, not indirectly as in PaaS or CMP options.

Having said that, the three options may not always be mutually exclusive, and it is also common to see a combination of some of the tools. For example, a generic Orchestration tool can be used to configure and setup a PaaS like Cloud Foundry or Kubernetes. A CMP can have an Orchestration framework as an add-on service that runs through the CMP thereby providing both the application and infrastructure view combined.

GIGASPACES

www.getcloudify.org

GigaSpaces Offices Worldwide

US East Coast Office, New York
Tel: +1-646-421-2830

US West Coast Office, San Jose
Tel: +1-408-816-1740

International Office, Tel Aviv
Tel: +972-9-952-6751

Europe Office, London
Tel: +44-207-117-0213

Asia Pacific Office, Hong Kong
Tel: +852-37198212