Cloud Management in the Enterprise

An Overview of Orchestration vs. PaaS vs. CMP

By: GigaSpaces Research, Cloudify Team
Contributors: Nati Shalom

Orchestration vs PaaS vs CMP

Table of Contents

Introduction....3

Categories For Comparison....4

Orchestration Defined....6

PaaS Defined....9

CMP Defined....10

Application Delivery....11

The Key Benchmarks....11

Conclusion....12

Introduction

Most top tier organizations today have been struggling with the aggregation of too much technology and tooling, fragmented versioning, and often times diverse tool sets across different business units in silos. Complexity is then added when initiatives of migration to the cloud are dictated from above, and managing these environments and applications in dynamic and elastic environments often times becomes a real pain point.

Many tools and approaches have arisen to help tackle this issue, however the prevalent approaches for managing applications in cloud environments usually fall under the categories of orchestration, PaaS (Platform as a Service) and CMP (Cloud Management Platform). The end goal of all three disciplines is fairly similar, they are intended to provide a manner to fully manage applications on the cloud. It is therefore not surprising that those looking to solve application management in the cloud have a hard time choosing which of the approaches best suits their needs, and differentiating between the benefits of each.

This white paper will make a simple distinction between these three options and offer some guidance on when to consider each of the different approaches by presenting popular use cases and scenarios.

Categories for Comparison

While there are an endless number of categories that can be compared between the different technology approaches, this paper will focus on the categories below that will serve as the basis for the comparison of the three different cloud management approaches.

1. Application Workloads

This category compares the three approaches based on the kind of application workload that best fits each platform, for example: web applications, stateful applications, big data, legacy, based on the ability to run and manage them effectively on the selected platform, which is mission critical for enterprises these days as these types of workloads represent the lion's share of enterprise workloads today.

2. Support for Advanced IaaS Services

This category compares the three approaches based on the depth of IaaS support, for example: platform utilization of advanced IaaS services such as DBaaS, LBaaS, EMR, among others.

3. Support for DevOps Processes

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

4. Support for Containers and the 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 diverse container technologies in that context e.g. Docker Swarm, Kubernetes, Mesos, Fleet.

5. 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 deployments 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 will compare the three approaches based on their support for bare metal environments.

6. Support for Network Orchestration

Many enterprises and clouds support virtual networking through SDN (software-defined networking). In this category we compare the three approaches based on the degree of support for network orchestration, a critical aspect in enterprise IT for connecting the HW and SW layers, as well as security and many other considerations.

7. 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 IT processes that have formerly been performed manually, such as provisioning, installation, configuration management, remediation, maintenance, monitoring, scaling, et al. This is largely achieved by modeling applications into "blueprints" that describe the application's topology, dependencies, workflows, and triggers for maintenance and remediation - enabling fine-grained management of the entire lifecycle of the application from provisioning through post-deployment. There are infrastructure-centric orchestration tools, such as OpenStack Heat and AWS CloudFormation, alongside container orchestration tools such as Docker Swarm and Kubernetes, and there are open source tools such as Cloudify.

- Application Workloads:
  The automation approach is fairly unopinionated and therefore can address any use case that can be mapped into distinct manual steps by scripting them into the templated blueprint. 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. This means that the underlying infrastructure's built-in services are available even as they rollout, and these often times include services that are mission critical for enterprise applications, including load balancers as a service, firewalls as a service and more.

- Support for DevOps Processes:
  As orchestration models the application into blueprints, it is 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 they are basically execution scripts. Workflows can easily interact with any external system such as a build system or support system as part of the continuous deployment process by simply adding them as a "type" or "input".

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

  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 container technologies such as

Kubernetes and Mesos, alongside non-container stack technologies and stateful services 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

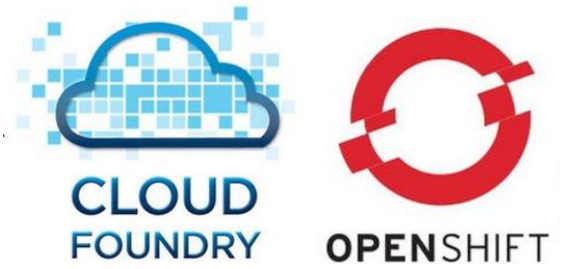
PaaS Defined

Definition

PaaS (Platform as a Service) was built as an abstraction layer on top of IaaS, that is aimed to help the developers focus on writing code by abstracting all the infrastructure and operational aspects, and takes a more a developer-centric approach, as a result. 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). There are infrastructure-centric PaaS, such as AWS Elastic Beanstalk, closed-source PaaS like Heroku, and the open source versions such as Cloud Foundry and OpenShift.

- **Application Workloads**:
  PaaS products are mostly suited for greenfield applications that fit the 12-factor approach. PaaS are meant to enable you to deploy applications directly from your IDE, and not have to worry about configuring the underlying infrastructure. 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 byproducts that comes with this kind of 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 that are important for enterprise deployments. 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 the 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 and what it exposes.

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 CoreOS 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. For example, load balancers, firewalls, vLANs that are for the most part pre-defined.

- Target Users:
  Developers

Additional Reference: PaaS vs Orchestration - PaaS vs Docker

CMP Defined

Definition

CMP (Cloud Management Platform), take an infrastructure-centric approach where the main focus is monitoring and managing of infrastructure resources such as virtual machines, storage, networks. It can be used indirectly to manage applications by combining some orchestration capabilities as part of the platform. There are proprietary cloud management platforms such as RightScale and VMware vROps, alongside open source platforms such as CloudForms.

- **Application Workloads**:
  Similar to orchestration, 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.

  Automating the full lifecycle of a given application, including configuration management and automation of post-deployment aspects such as failover 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 – such as metering and chargeback. To achieve this goal, many of the CMPs have to force some degree of the "least common denominator" approach to 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 times not a trivial task, 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, et al 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 the Cloud Native Stack**:

Most CMPs can run containers on top of VMs, but usually the integration is fairly loose and requires more external configurations and intervention for handling areas such as scaling, integration with other systems, automating the actual deployment in an infrastructure-centric approach that doesn't always take the applicative aspects into consideration.

- 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 provide some insight into an area that often times has blurred lines regarding where one ends and the other begins, as well as what services are supported by each set of tools, making it difficult to compare which tooling is the right one for the job.

With the growth of cloud and the need to manage complex applications in elastic environments, the parameters chosen for comparison are those often encountered in large enterprises when considering the management of applications in 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 developers need to be provided with the right tools for the job. Every developer may have different needs and, therefore, tools needed to serve the applications, see: "What Developers Want".

The Key Benchmarks

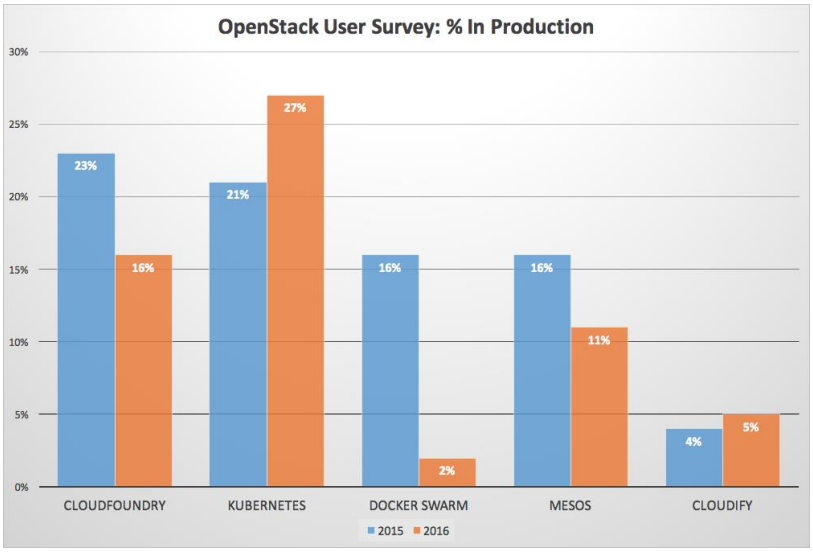
Based on the above analysis, each approach can be measured based on the following criteria:

- How fast can tools be served to developers that need to become productive?

The open-source world offers plenty of new frameworks and tools that are announced almost on a daily basis. Developers want to have access to them all, as, in many cases, these 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 these 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.

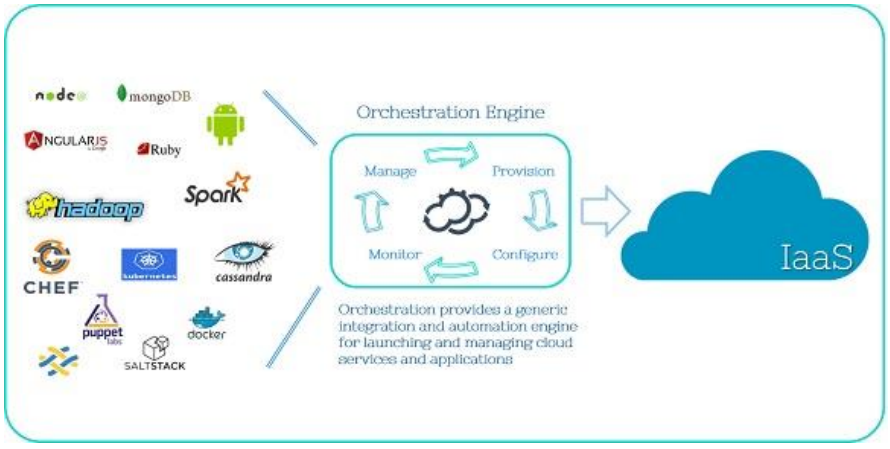


As we can see in the following OpenStack surveys from 2015 and 2016, the popularity of the various platforms tends to change rapidly. In this specific case, we can see that Kubernetes grew almost 30% at the expense of Cloud Foundry and Docker's own tool, Swarm. Cloudify grew 20% in use with OpenStack 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 you are at managing many different types of frameworks?

Based on this, it is apparent that in order to achieve this goal there needs need to have a generic way enables any application and framework to be taken and offered as a managed service.



Conclusion

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, and more – directly, not indirectly as with PaaS or CMP options.

That said, 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.

A common trend that is now becoming apparent is orchestration tooling as the next-gen CMP. In this way, we gain an application-centric view of the world coupled with the ability to manage and monitor the infrastructure level with a direct correlation to the application usage which provides more business value for IT at an enterprise scale.

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