Practical Guide to Cloud Deployment Technologies

Version 1.0

A Discussion Paper from the OMG Cloud Working Group

March 2019

Document mars/2019-03-03

This paper presents a discussion of technology issues considered in a Subgroup of the Object Management Group. The contents of this paper are presented to foster wider discussion on this topic; the content of this paper is not an adopted standard of any kind. This paper does not represent the official position of the Object Management Group.

This page intentionally left blank

Table of Contents

Acknowledgements

Development of the Practical Guide to Cloud Deployment Technologies is a collaborative effort that brings together diverse customer-focused experiences and perspectives into a single guide for cloud customers. The following participants have provided their expertise and time to this effort:

- Machiel Andeweg (IBM)
- Claude Baudoin (cébé IT & Knowledge Management)
- Jeff Boleman (IBM)
- Christian Boudal (IBM)
- Salvatore D'Angelo (University of Campania)
- Hannah Day (Mayo Clinic)
- Beniamino Di Martino (Second University of Naples)
- Chris Dotson (IBM)
- Ellen Feaheny (AppFusions)
- Jean-Claude Franchitti (Archemy)
- Rajesh Jaluka (IBM)
- Srinivasa Reddy Karri (Schlumberger)
- Salvatore Augusto Maisto (University of Campania)
- Stefania Nacchia (University of Campania)
- Mario Panagakis (IBM)
- Massimilliano Rak (Second University of Naples)
- Karolyn Schalk (IBM)
- Lisa Schenkewitz (IBM)
- Karl Scott (Satori Consulting)
- Sri Shetty (IBM)
- Nancy Smallwood (Vertex)

Executive Overview

The objective of this guide is to define and position the cloud technologies that can be used to deploy cloud-based applications and services, and clarify how they differ in their implementation and use. There are many options available, and there is no right or wrong choice. Many organizations will need to implement more than one option. Accordingly, selecting the best option to support workloads while controlling complexity can be a daunting task. This practical guide will help developers, architects, and IT leaders make informed decisions and select the best technology for their specific needs. These technologies span storage, network, and compute services. This Practical Guide will focus on compute options.

What is a Cloud Deployment Technology?

This section defines the different cloud deployment technologies and why they are important. This includes describing application dependency on cloud deployment technologies and why workloads (e.g., high-performance computing [HPC], transaction computing, and analytic computing) require different solutions.

Cloud computing has transformed and continues to evolve the delivery of technology services. Capabilities such as broad network access, measured service, multi-tenancy, rapid elasticity and scalability, and resource pooling have enabled innovative services. The characteristics and benefits of cloud computing are described in more detail in the *Practical Guide to Cloud Computing* [1]. The continual evolution of cloud deployment technologies introduces new possibilities to enhance the efficiency of cloud services.

Cloud deployment is traditionally defined in terms of service models and deployment models. The "classic" definition of cloud service models comes from the *National Institute of Standards and Technology (NIST) Special Publication 800-145* [2], which defines three service models:

- Infrastructure as a Service (IaaS)
- Platform as a Service (PaaS)
- Software as a Service (SaaS)

NIST SP 800-145 also defines four deployment models:

- Private cloud
- Community cloud
- Public cloud
- Hybrid cloud

Figure 1 describes the relationship between cloud deployment models, service models, and cloud deployment technologies. When considering workload and service deployments, there are three major areas of architectural consideration. The first is referred to as **cloud deployment model options**, the second is **service model options,** and the third is **cloud deployment technology.** Deployment model options are normally considered first or at the same time as the service options, depending on requirements. The deployment model options determine whether the project will be hosted in a public, private or hybrid cloud environment. The service model options determine the level and type of service (e.g., IaaS and PaaS) chosen from the cloud provider.

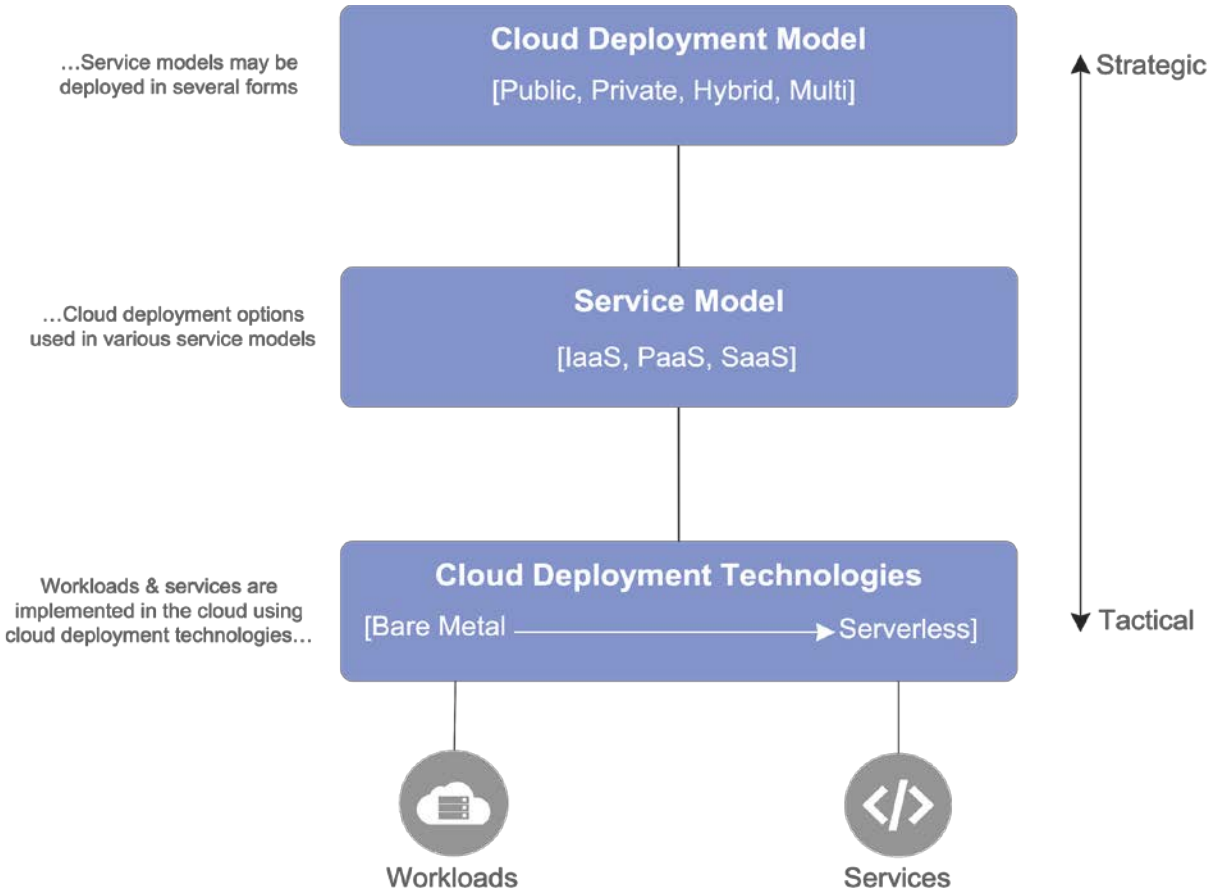

Figure 1 – Workload and Service Cloud Deployment

Cloud deployment technologies are used to place workloads into a cloud computing environment. They provide the foundation to facilitate execution of applications and services. These technologies range from bare metal to serverless computing and enable organizations to realize the benefits of cloud computing. The Definition of Cloud Deployment Technology Options section of this Practical Guide provides insight into the available technologies.

Customers must understand the cloud deployment options that are available and leverage the best option to ensure the performance, speed of deployment, elasticity, portability, and security needed to support the business needs while containing cost. The variety of workloads and services supported by the enterprise requires different considerations when deploying them in a cloud environment. For example, HPC workloads are compute- and memory-intensive, while a legacy application may require significantly less resources.

Navigating the many cloud deployment technology options and establishing a plan to manage adoption is required to contain complexity. The next section describes the cloud deployment technology options in detail and how they fit within various service models.

Definition of Cloud Deployment Technologies

Cloud deployment technology options have evolved rapidly over recent years. Virtualization in the form of virtual machines (VMs) was initially used to deploy workloads in a cloud environment. This level of abstraction was widely accepted as it provided isolation at the operating system level and allowed the administrator to maintain a level of control. The need to increase the speed of service deployment along with the reluctance to manage systems increased the adoption of serverless computing.

Currently, there are four technology options: **bare metal, VMs, unikernels,** and **containers.** Unikernels, while viable, apply to a narrow set of data center use cases and have not attained wide adoption. Cloud service providers offer the other three options (bare metal, VMs, and containers) in various service models: Infrastructure as a Service (IaaS), Container as a Service (CaaS), Platform as a Service (PaaS), Function as a Service (FaaS)/Serverless. As described in Figure 2, PaaS and FaaS/Serverless may leverage IaaS or CaaS for deployment.

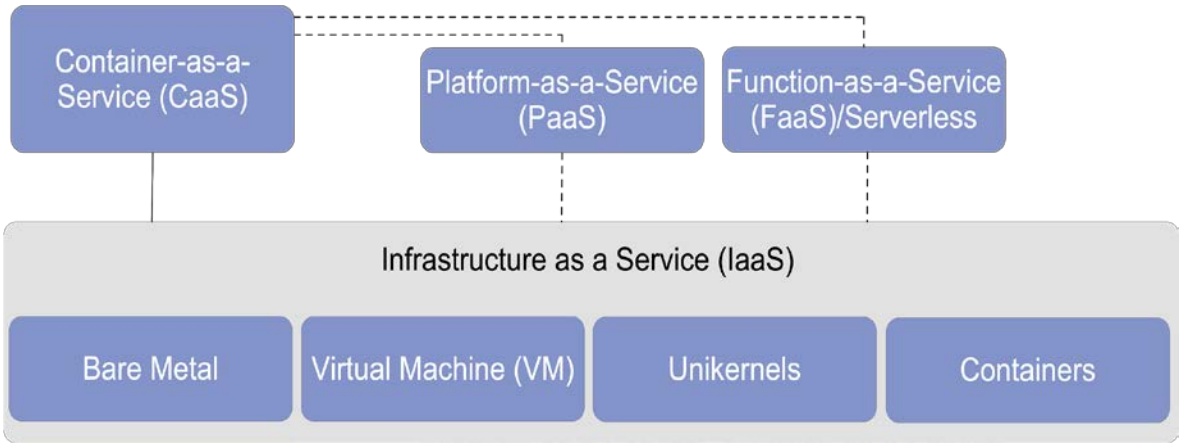

Figure 2 -- Cloud Deployment Technology Options

Cloud providers are constantly innovating and coming up with "X as a Service" (XaaS) models such as Disaster Recovery as a Service (DRaaS), Backup as a Service (BaaS), Business Process as a Service (BPaaS), and Desktop as a Service (DaaS). This Practical Guide focuses on IaaS and the higher-level container, platform, and Function as a Service options used to deploy cloud workloads and services.

IaaS Deployment Options

Bare metal servers, VMs, unikernels and containers are made available to the customers in an IaaS model. These deployment options offer a level of infrastructure control to address specific performance requirements, compliance considerations, and legacy dependencies. This level of control also brings responsibility to manage the infrastructure.

Bare Metal

A **"bare-metal server"** is a high-performance cloud server that is composed of a single-tenant physical server. On a bare metal server, the operating system can be installed directly on the server without the use of a hypervisor, eliminating layers and delivering better performance. These servers may run any amount of work for the customer, or may have multiple simultaneous users, but they are dedicated entirely to the customer.

VMs

VMs utilize software such as hypervisors to enable multiple servers to operate on the same physical system. The hypervisor is a software component that allows multiple "guest" VMs to be installed and run while maintaining isolation. Each VM has its own operating system, kernel, and allocation of resources (e.g., CPU and memory). VMs enable cloud customers to run multiple servers on a single physical computer, thereby maximizing the utilization of resources and reducing cost.

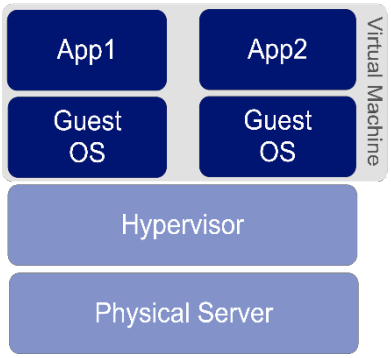

Figure 3 – VM Architecture

Unikernels

Unikernels represent a different approach to reduce the server footprint needed to support applications. They are specialized, single-purpose machine images constructed by using a dedicated kernel with only the libraries needed to run the application. This kernel architecture was originally introduced as MirageOS.

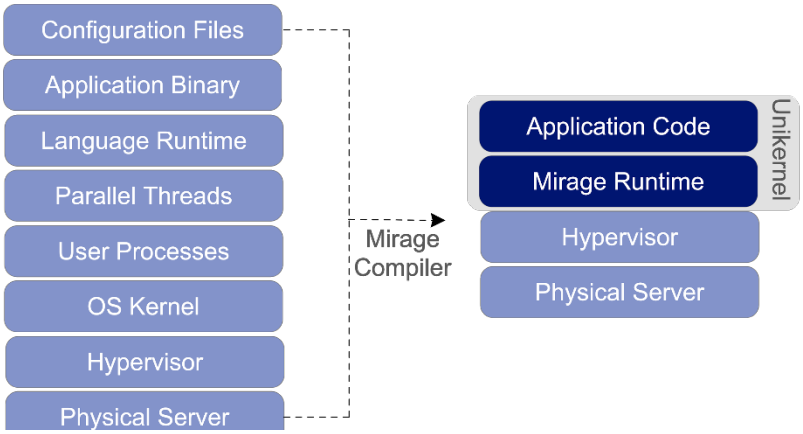

Figure 4 -- Unikernel Architecture

Unikernels are built by compiling high-level languages directly into specialized machine images that run directly on a hypervisor, such as Xen, or on bare metal. The unikernel architecture has minimal overhead and can start very quickly—sometimes in milliseconds. That makes them faster than containers, which usually take at least a few seconds to spin up, and much faster than VMs, which can take minutes. Additionally, some consider unikernels more secure due to their smaller attack surface.

The adoption of unikernels remains low due to a combination of lack of awareness and low availability of orchestrators. They are becoming more prevalent in single purpose devices such NFV (Network Function Virtualization) appliances, but data center adoption remains low.

Containers

Containers are not a virtualization technology; they are a standard way of packaging code, configuration, and dependencies into a single deployable image. This makes the image portable across different operating system, and enables it to run in an isolated working environment. The isolation makes it possible to run many containers on a single bare metal or VM without any interference between them. This can help significantly reduce the number of compute instances needed to run applications. Containers with different base OS can run on the same server that may run a different operating system. All containers that are running on a host share the same kernel, and that kernel manages service isolation such as separate process lists, users, or network interfaces.

Although containers can be used like small VMs, this is not the best way to use them. Containers should generally hold the bare minimum operating system components needed to perform their function, and each container should only perform a single function.

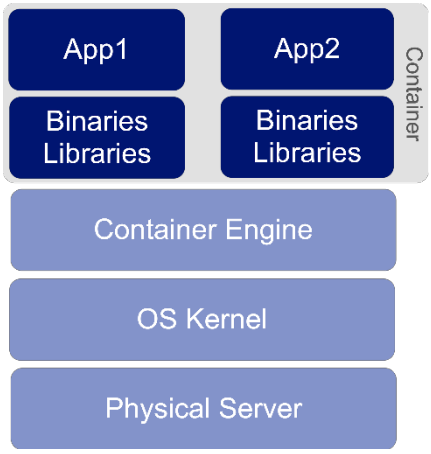

Figure 5 – Container Architecture

Container deployment options include building your own container service, utilizing off-the-shelf CaaS platforms, or choosing managed container services, such as managed Kubernetes, provided by large Cloud Service Providers (CSPs).

- In the build-your-own-service option, customers build and deploy their own containers on top of IaaS, either bare metal or VM. While the cloud service provider is responsible for managing the underlying infrastructure, customers are responsible for installing container technology; setting up scheduling, orchestration, and cluster management; and creating container images.
- When using an off-the-shelf CaaS platform, the cloud customer may choose to deploy it on-premises or in a public (or off-premises private) cloud environment. The customer is responsible for containerizing the middleware, libraries, and applications or microservices. The overhead for the customer is higher when compared to PaaS, because they are responsible for creating and updating the container images for each component (e.g., language and middleware).
- When using a managed service, the CSP manages the container engine and underlying infrastructure, while the customer manages the workloads.

Platform as a Service (PaaS)

PaaS is a service delivery model in which the cloud service provider pre-packages middleware, language runtimes, and tools as containers. The provider is responsible for managing the container image and underlying infrastructure on which the container images are provisioned. This significantly reduces the overhead for the developers, enabling them to focus on developing and deploying code.

PaaS is offered in a hosted (public or private) or installable (on-premises) form. This enables the customer to select the best deployment option to fit their needs.

Function as a Service (FaaS)/Serverless

FaaS and Serverless computing are synonymous [3]. FaaS/Serverless computing enables customers to run server-side software in ephemeral, stateless containers without the need to manage the underlying servers. FaaS/Serverless applications have the following basic characteristics:

- Event-based architecture – code runs in response to some event. The code may initiate other events, which could trigger further code to run.
- The unit of deployment is the code and configuration to run in a managed environment.
- Functions running in FaaS are stateless.

The FaaS/Serverless solution monitors load and automatically scales the underlying infrastructure to satisfy demand. Customers must be aware that providers place limits on the resources available to functions, processing time, and concurrency of event processing. These constraints must be assessed to ensure the user experience is not degraded.

FaaS/Serverless and PaaS Comparison

FaaS/Serverless and PaaS, on the surface, appear to be very similar in their implementation and use from a development point of view. In reality, these two cloud deployment services are quite different. Scalability, pricing, startup time, tooling, and the ability to deploy differ between the two platforms.

Serverless applications offer true auto-scaling capabilities to customers without any additional configurations. Developers must forecast resource capacity and configure PaaS-hosted applications to scale up and down based on the demand. Serverless cost structure is based on usage, and there are no fixed monthly charges for services. In some cases, serverless can be a more cost-effective option for software deployments.

Container startup time in a FaaS/Serverless environment is much faster than PaaS. Fast startup is critical to satisfy the needs of event-based processing. PaaS is tolerant of slower startup times because the applications supported on the platform are long-lived.

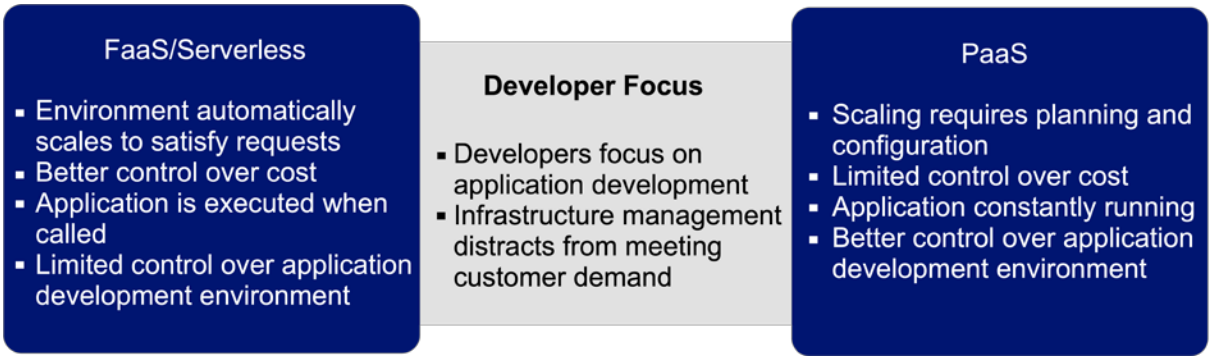

Figure 6 – PaaS and Serverless Comparison

The developer has more control of the application development environment when working in PaaS. FaaS/Serverless IDEs (Integrated Development Environments) must support the target platform (e.g., AWS Lambda or IBM OpenWhisk); limited provider agnostic frameworks are available. Additionally, the developer is constrained by a smaller set of development languages. These limitations impact the level of application development control within a FaaS/serverless environment.

The available cloud deployment technology options will continue to evolve. The options referenced in this section are used by many cloud customers to support disparate workloads and services. Insight into the use cases associated with the options would help determine how they apply within the environment and software development houses.

Cloud Deployment Technology Use Cases and Implementation

This section describes the uses cases associated with the cloud deployment technologies referenced in the previous section of this document. The use cases are focused on the most common data center deployments within the enterprise and software development houses. There are additional considerations for technology use in embedded devices and edge computing; those are outside the scope of this *Practical Guide.*

Use Cases

Multiple cloud deployment technologies may be used to deploy applications and services to the cloud. For example, a combination of bare metal, PaaS, and FaaS may be required to support a large-scale e-commerce application. Cloud customers must assess the application architecture, data implications, runtime dependencies, and constraints before selecting the right cloud deployment technology to satisfy their needs.

Table 1 highlights the uses cases for the various cloud deployment technologies. The table is followed by a more in-depth discussion.

Table 1 – Use Cases for Each Cloud Deployment Technology

Technology	Use Cases
Bare Metal	• Migrate legacy applications to the cloud • Deploy HPC services • Deliver cloud-based latency sensitive applications • Need for very large relational database in the cloud • Comply with regulations or contract agreements that require single tenant hosting • Need for Trusted Platform Modules (TPM)
VMs	• Migrate legacy applications to the cloud • Deploy monolithic or tiered applications • Enhance container security
Containers / CaaS	• Deploy microservice-based applications • Deploy legacy applications on operating systems that are no longer supported • Immutable infrastructure • Service flexibility
Unikernels	• VM optimization • Immutable infrastructure
PaaS	• Deploy microservice-based applications • Enhance developer productivity • Reduce infrastructure responsibility
FaaS/Serverless	• Deploy microservice-based applications • Support event-driven processing • Improve service elasticity

Bare Metal

The capacity of bare metal servers and lack of contention for their resources (CPU, memory, and I/O) makes this a viable option to support resource-intensive workloads such as HPC. This includes latency-sensitive applications (e.g., streaming services) and large databases that consume significant resources. Customers may combine the use of bare metal servers for the resource-intensive components of the solution architecture with VMs, containers, and other cloud deployment technologies to deliver application services.

Bare metal servers are also an option for legacy applications that may not be virtualized or not well suited for a virtualized environment.

VMs

Legacy applications that can be virtualized but have known behavioral issues that can be managed are well-suited for cloud deployment using VMs. This also applies to legacy applications that are not well documented but there is basic knowledge of how they run.

Monolithic and tiered application architectures are not extinct. Thought leaders in software architecture promote these architectures for application designs that are not yet mature enough for a microservices architecture. VMs are a viable option for monolithic and tiered applications.

VMs can be used as an additional layer of isolation when deploying containers. This approach provides additional assurance for customers uncomfortable with a shared kernel model. There is additional overhead associated with the VM, but this can be justified by the enhanced security.

Containers/CaaS

Cloud customers have developed a plethora of microservices ranging from user interface (UI) frontends to backend microservices. Containers enable scaling, replicating, and choreographing services in a microservice architecture. The ability to start containers in a fraction of the time of VMs is critical. This fast startup enables adherence to a key *Twelve-Factor App* [4] principle: disposability.

Legacy applications that depend on out-of-support operating systems are candidates for containers. Application dependencies can be containerized to enable execution. There are limitations to this approach, however. The application dependencies must be assessed to ensure containerization is a viable option.

Platform flexibility is the primary use case for CaaS. Customers may not want to be locked into a PaaS due to the restrictions on development languages and limitations on platform tools. On the other hand, the complexity and cost of managing containers is prohibitive. CaaS provides the customers with a container-based environment they can control (security, scalability, and availability) without significant effort, and does not limit them to the languages, middleware, and tools offered by the PaaS platform.

Unikernels

The need to optimize VMs is the most common use case for unikernels at this time. This includes reducing image size to improve start time and improve security. Removing unneeded binaries and libraries result in a much smaller image that can be provisioned and started more quickly. The absence of the additional components reduces the cybersecurity attack surface, thereby lowering risk. Additionally, the increased isolation from the host provides an added level of security.

Immutable infrastructure is a concept pursued by some cloud customers to enhance the integrity of the environment. For example, changes to VMs are not permitted in an immutable environment. To upgrade an application, the current instance is destroyed and a new instance containing the updates is

provisioned. In this scenario, the cloud customer knows exactly the state of the environment. Unikernels promote this concept of immutable infrastructure.

PaaS

PaaS is used to deploy modern, cloud-native application architectures. PaaS is most often delivered using a container-based infrastructure, and therefore supports the same use cases as containers.

The most prevalent use case for PaaS is speed to market. PaaS reduces or eliminates the burden of deploying and managing infrastructure, enabling cloud customers (as well as software development houses, which often do not have the budget to hire personnel to build and manage infrastructure) to focus instead on customer needs and development of the applications. PaaS provides a development and CI/CD (Continuous Integration/Continuous Deployment) environment to quickly develop, integrate, build, and deploy applications.

FaaS/Serverless

The use case for FaaS/Serverless is limited to microservice-based applications, more specifically applications that are short-lived and event-based. Examples of services include image processing, reservation handling, and IoT data processing. Rather than running a server to satisfy requests, a function can be triggered by requests from an API gateway.

FaaS/Serverless is used to reduce the customer's responsibility to manage middleware dependencies and runtime considerations. The FaaS/Serverless platform takes care of this, leaving the customer with the responsibility to write the function code. The result is increased velocity in delivering technology capability and satisfying business demand.

Implementation

The intent of this section is to describe the tools cloud customers must consider when implementing services using the cloud deployment technologies presented in this Guide. Cloud customers who neglect automation will experience higher costs and slower deployment speeds. These tools are required to automate tasks such as server provisioning and configuration management. Table 2 lists such tasks that must be automated in the case of each cloud deployment technology.

Table 2 -- Tasks to Automate with each Cloud Deployment Technology

Cloud Deployment Technology	Tasks to Automate
IaaS	Orchestration and provisioning
PaaS	Source code management and CI/CD to deploy application artifacts on to a PaaS runtime platform
FaaS/Serverless	Orchestrating deployment of functions and dependencies onto a FaaS/Serverless runtime platform.

Automating the tasks listed in Table 2 improves the efficiency of a DevOps team and speeds up the deployment of services. Table 3 provides a list of tools available to assist with automation of development, delivery, and management of applications. This is not an exhaustive list of solutions available in the marketplace.

Table 3 – Examples of Automation and DevOps Tools

Tool Categories	Tool Names
Orchestration	Ansible, Pulumi, Terraform
Configuration Management	Ansible, Chef, Puppet, Salt
Source Control Management	BitBucket, GitHub, GitLab
Continuous Integration	Bamboo, CircleCI, Codeship, Jenkins
Build	Broccoli, CircleCI, Codeship, Maven
Continuous Deployment	Codeship, Go, Julu, Octopus Deploy
Operational Intelligence	Datalog, Dynatrace, New Relic, Prometheus
Collaboration	Jira, Slack, Trello

The next subsections expand on the information in Table 2 by describing in more depth the role of automation tools within each deployment technology.

IaaS

A new way of working is required when adopting IaaS. *Infrastructure as Code* [5] removes manual steps such as operating system installation, kernel modifications, and application configuration; it automates them using software code that references objects such as templates and recipes.

Orchestration and configuration management tools provide foundational capabilities to automate services using bare metal, VMs, containers, and unikernel cloud deployment technologies:

- Orchestration tools allocate server, network, and storage resources needed to support an application or service
- Configuration management tools configure what is inside a server, network, or storage device.

Each technology has intricacies to consider when automating – those details are outside the scope of this Practical Guide. CAML (Cloud Application Modeling Language) [6] facilitates the definition of the resources needed to deploy services in the cloud.

Orchestration and configuration management tools support a wide variety of IaaS providers (e.g., Amazon Web Services, Google Cloud Platform, IBM Cloud, and Microsoft Azure) and cloud deployment technologies. Some technologies such as unikernels are not widely supported by all tools and make adoption more challenging.

PaaS

The tools made available by a PaaS provider – whether open-source or proprietary – greatly eliminate manual tasks in the software development process. PaaS providers integrate those tools to streamline the code integration, build, and deployment process.

Infrastructure as Code can also apply in the PaaS environment. For example, the cloud customer may also need to provision resources and configure containers to support the applications and services hosted on the PaaS platform.

FaaS/Serverless

Serverless computing implies that server maintenance is not required. However, setup of serverless runtime is required. Deployment of functions, cloud service dependencies (e.g., S3 storage), and variable assignments (e.g., memory size) must be addressed. Tools such as Ansible include the capability to deploy cloud services in such an environment.

In addition to the tools described above, cloud service customers must consider cloud management platforms (CMPs) to simplify the deployment of services across multiple cloud service providers. The *OMG Practical Guide to Cloud Management Platforms* [7] provides insight into these platforms.

Understanding the available cloud deployment technology options and their most common uses cases informs the selection process. Cloud customers manage diverse, complex environments with many applications and services. Aligning the cloud deployment technology to ensure effective and efficient delivery is key to enabling digital services.

Selecting Cloud Deployment Technologies

This section describes the key considerations for selecting the right cloud deployment technologies. The selection process is part strategic and part tactical. Strategic cloud deployment technology selection considers the business needs and corporate cloud strategy, and determines which cloud deployment technologies are relevant. These decisions have a broad impact. Tactical selection addresses specific application/service needs and have a narrower impact.

The steps associated with selection include:

- Assess the fit of cloud deployment technologies to the business needs
- Understand workload/service requirements
- Select application/service cloud deployment technology

Each of these steps is described below.

Assess the Fit of Cloud Deployment Technologies to the Business Needs

Cloud customers should identify which cloud deployment technologies meet the needs of their business. This exercise should be performed again periodically since new options regularly appear. The key considerations in the selection process are as follows.

1. **Cost -** Identify the cost model, cost drivers, and variables associated with each cloud deployment technology.
2. **Architectural fit –** Understand the application architectures that will be supported. The ability to support monolithic and tiered applications must be considered.
3. **Performance –** Determine the performance requirements associated with the workloads. High-bandwidth, low-latency applications may require bare metal deployment, while VMs may be sufficient for internal-facing applications with less demanding resource utilization.
4. **Compliance –** Identify significant regulatory and customer contractual limitations that impact the level of isolation required. This information, combined with an understanding of sensitive data flows, helps determine the right cloud deployment technology.
5. **Elasticity requirements –** identify the level of elasticity needed to satisfy the current and forecasted workloads. The cloud deployment technology must be able to support the variability in processing and target service levels for service startup.
6. **Control requirements –** Determine the organization's need to control layers of the stack. Customers that require great control may focus on VMs and containers. Customers requiring limited control may select PaaS and FaaS/Serverless services.
7. **CSP lock-in –** Determine the level of portability needed to enable flexibility and to manage the risk of poor provider performance, cost increases, and other considerations, by having the option to change provider. A broader choice of offerings supporting the solution architecture will decrease the risk of lock-in.

Most often, selection of cloud deployment technologies is an informal process. Formalizing this process allows the customer to implement the appropriate orchestration and management tools, and establish processes needed to deliver effective and efficient services.

Understand Workload/Service Requirements

The *Practical Guide to Hybrid Cloud Computing* [8], in the section entitled *Cloud Deployment Model for Applications and Data*, provides guidance to identify requirements related to the types of workloads or services being deployed in the cloud.

Those requirements include meeting constraints such as:

- strict latency limitations
- data protection controls associated with sensitive personally identifiable information (PII) or patient health information (PHI)

Each industry sector and type of application may have such specific requirements.

Select Application/Service Cloud Deployment Technology

After defining the strategic (business-oriented) requirements and the technical constraints (workload- and service-related), the customer can rate each cloud deployment technology option vs. the requirements, and make a decision. The selection criteria used to reach a decision are those from the previous two subsections – business needs and workload/service requirements.

Selecting the right cloud deployment technology is critical to adhering to service levels and satisfying business demand. Proactively investigating the available technologies and defining the selection criteria simplifies the selection process, and allows to repeat it periodically as the range of available technologies continues to increase.

Applications Composed of Multiple Cloud Services

An increasing number of use cases require combining multiple cloud applications or services with shared data sets and integrated functionalities referred to as “cloud mashups.” For example, AWS EC2, Facebook’s authentication and authorization services, and Google’s MapReduce service can all be mashed up to provide a driving route recommendation service.

Proper selection of cloud deployment technologies is also important for such multi-cloud mashups. On one hand, cloud deployment technology influences the overall quality of composite web services in intercloud applications. In this context, quality is typically measured in terms of Quality of Service (QoS) and Quality of Experience (QoE) assurances, where QoS and QoE specify the desired performance requirements in the mashup services (e.g., driving route recommendation accuracy and speed of route calculation in our example). On the other hand, cloud deployment technology choices can help streamline the subsequent discovery of best-choice web services and facilitate the otherwise complex creation (i.e., dynamic composition) of mashup services as integrated packages for users.

There is an increasing demand for such web/cloud services that provide personalized, cognitive applications aimed at improving the user’s life. As a result, interest in mashup service discovery and automated composition techniques will keep increasing. Cloud solutions providers already compete to satisfy requests for mashup services. Emerging marketplaces for reusable cloud components provide APIs and mashups (e.g., Programmable Web, Consensys/Ethereum marketplace). The proper selection of cloud deployment technologies will help create more agile and scalable cloud mashups.

Summary of Keys to Success

The range of cloud deployment technologies is broad (bare metal to serverless) and the number of options will continue to grow. Cloud deployment technologies can be confusing as some technologies overlap and other options are services in disguise (e.g., PaaS, FaaS/Serverless). Understanding the options available and how they can satisfy the needs of legacy and modern applications and services is vital when deploying services to the cloud. Table 4 describes the keys to successfully utilizing cloud deployment technologies to deliver disparate applications and services.

Table 4 – Keys to Success

Key Factor	Description
Understand the Range of Cloud Deployment Technologies	• Understand where cloud deployment technologies fit in the delivery of cloud services • Identify the available cloud deployment technologies and associated services • Understand how the various cloud deployment technologies work, along with the differences between overlapping services
Determine Implementation and Support Options	• Determine the appropriate use cases for available cloud deployment technologies • Identify the tools required to support delivery of services – automation is critical
Establish Criteria and Select Cloud Deployment Technologies	• Assess the fit of cloud deployment technologies to the business needs • Understand the technical factors – application architecture, data implications, runtime dependencies • Select the “best fit” application/service cloud deployment technology

Customers must periodically re-asses the respective value and fit of various cloud deployment technologies to their business. While multiple options may coexist (one size does not fit all uses), mixing and matching all available options can be a waste of productivity. The effort spent researching and selecting the right solutions upfront has a significant return on investment when compared to the cost overruns, suboptimal user experience, and increased risk that can result from an ad-hoc approach.

References[1]

[1] Cloud Standards Customer Council (2017): *Practical Guide to Cloud Computing, V3.0.* https://www.omg.org/cloud/deliverables/practical-guide-to-cloud-computing.htm

[2] National Institute for Science and Technology (2011): *Special Publication 800-145: The NIST Definition of Cloud Computing.* https://csrc.nist.gov/publications/detail/sp/800-145/final

[3] Roberts, Mike (2018): *Serverless Architectures*. Published on MartinFowler.com. https://www.martinfowler.com/articles/serverless.html

[4] Wiggins, Adam (2017): *The Twelve-Factor App*. https://12factor.net

[5] Morris, Kief (2016): *Infrastructure as Code.* O'Reilly Publishing. ISBN 978-1491924358. http://infrastructure-as-code.com/

[6] Bergmayr, A., J. Troya, P. Neubauer, M. Wimmer, and G. Kappel: *UML-based Cloud Application Modeling with Libraries, Profiles, and Templates.* Proceedings of the 2nd International Workshop on Model-Driven Engineering on and for the Cloud (CloudMDE 2014). http://ceur-ws.org/Vol-1242/paper7.pdf

[7] Cloud Standards Customer Council (2017): *Practical Guide to Cloud Management Platforms.* https://www.omg.org/cloud/deliverables/practical-guide-to-cloud-management-platforms.htm

[8] Cloud Standards Customer Council (2016): *Practical Guide to Hybrid Cloud Computing*. https://www.omg.org/cloud/deliverables/practical-guide-to-hybrid-cloud-computing.htm

[1] References [1], [7] and [8] are to documents from the Cloud Standards Customer Council (CSCC), a program of the Object Management Group that was transitioned to the OMG's Cloud Working Group (OMG CWG) in 2018. Past CSCC documents are now available from the OMG CWG web site as indicated above, but are still under CSCC cover. They will be rebranded as OMG documents as they get revised in the future. New guides and papers, such as this one, follow the OMG brand and naming/numbering conventions.

