Cloud
Standards
Customer
Council

Practical Guide to Platform-as-a-Service
Version 1.0

September, 2015

Contents

Acknowledgements

The major contributors to this whitepaper are: Claude Baudoin (cébé IT & Knowledge Management), Michael Behrendt (IBM), Mike Edwards (IBM), Chris Ferris (IBM), Daniel Krook (IBM), John Meegan (IBM), Karolyn Schalk (Garden of The Intellect LLC), Joe Talik (AT&T), Steven Woodward (Cloud Perspectives).

Executive Overview

The aim of this guide is to provide a practical reference to help enterprise information technology (IT) managers, business decision makers, application architects and application developers understand the Platform-as-a-Service (PaaS) cloud service category and how it can be used to solve business challenges rapidly and cost effectively.

The cloud computing marketplace has grown rapidly to encompass a huge range of offerings. PaaS is one of the most dynamic areas of cloud computing, but there is some confusion about the definition of PaaS and the capabilities that should be expected of a PaaS offering. The *Practical Guide to Platform-as-a-Service* aims to define PaaS and differentiate it from other categories of cloud computing such as Infrastructure-as-a-Service (IaaS) and Software-as-a-Service (SaaS). The capabilities of PaaS systems are discussed and these capabilities have a direct bearing on the types of business problem and kinds of application that are best suited to use PaaS.

This Guide recommends best practices for using PaaS in terms of architecture, development processes, integration, deployment and operation. Differences between traditional application development and delivery and the appropriate techniques for PaaS platforms are highlighted, since in many cases the biggest gains for the enterprise result from the adoption of newer, more efficient, more rapid and less error-prone techniques for creating, testing and deploying applications.

One challenge that faces cloud service customers is that many different types of cloud service are given the label "PaaS", so that it can at times be difficult to evaluate what is being offered by the cloud service provider and even harder to compare offerings from different providers. In this Guide, the aim is to provide customers with an understanding of the range of capabilities that PaaS offerings provide – and to distinguish these from what is supplied by IaaS and SaaS cloud services. Inevitably, there are some fuzzy boundaries between the categories of cloud service and the Guide provides a discussion of these gray areas and how best to evaluate them.

The Guide is structured into three main sections:

- Setting the context for PaaS in cloud services – its characteristics and capabilities, the benefits of using PaaS, examples of PaaS offerings and matters of governance and business considerations when using a PaaS offering.
- Addressing the considerations relating to acquiring and using PaaS cloud services forms the central section of the guide, examining what types of applications are best suited to PaaS platforms, the architectural styles involved and the development and operations techniques best used.
- Finally, providing recommendations on how to best use PaaS services.

PaaS in the Context of Cloud Computing

Cloud computing is defined as a paradigm for enabling network access to a scalable and elastic pool of sharable physical and virtual resources with self-service provisioning and administration on demand [1]. The physical and virtual resources are offered as capabilities by cloud services, invoked through a defined interface. The resources include servers, data storage devices, networks, operating systems, software and applications.

Cloud computing offers almost all of the capabilities of IT systems through cloud services that are invoked across the network – and it is the sheer breadth of capabilities that can be a challenge for the cloud service customer to understand. The ISO/IEC cloud computing standards [1] [2] divide the capabilities offered by cloud services into 3 broad groups:

- Infrastructure capabilities, where the cloud service customer can provision and use processing, storage and network resources.
- Platform capabilities, where the cloud service customer can develop, deploy, manage and run applications (created by the customer or acquired from a third party) using one or more execution environments supported by the cloud service provider.
- Application capabilities, where the cloud service customer can use one or more applications supplied by the cloud service provider.

Unsurprisingly, PaaS concerns the provision of platform capabilities, but for better understanding it is worthwhile to examine all three groups of cloud services – and also the traditional approach of deploying applications in on-premises datacenters, as shown in Figure 1.

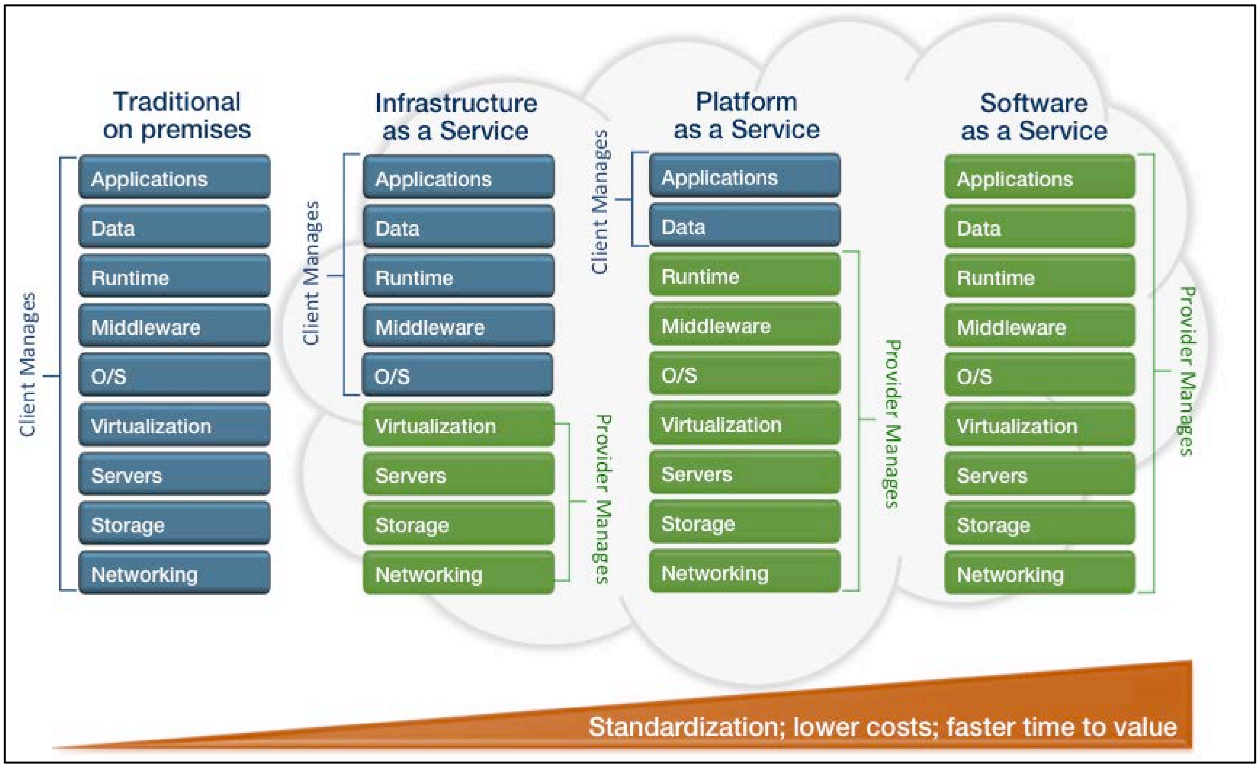

Figure 1: Application Deployment - Traditional and Cloud-based

It is important to note the focus on applications in this discussion. PaaS is primarily concerned with developing, deploying and operating customer applications – other capabilities may be involved, such as the use of processing, storage and network resources, but they are not the main focus.

Traditional On-Premises

In the case of deploying and operating applications in a traditional on-premises environment, the customer is responsible for acquiring, installing, configuring and operating all the elements of the ecosystem required to run the applications. This includes all the hardware components – servers, data storage, networks. It also typically includes an extensive software stack, starting with the operating systems (and hypervisors, if virtualization is employed) and various types of middleware and runtimes and finally the custom code of the applications themselves. There is also typically a host of supporting software, including databases, messaging systems, analytics programs, plus an extensive set of tools for management and monitoring of the application in production. In addition, software is required for the deployment and updating of the application software.

Infrastructure as a Service (IaaS)

Cloud services offering Infrastructure-as-a-Service provide basic IT resources – servers for processing, data storage devices and networking. IaaS systems typically provide virtualization, since this is the foundation for the sharing of resources that is the hallmark of cloud computing, enabling more efficient use of those resources and resulting in lower costs and higher flexibility.

The cloud service provider takes responsibility for and manages the resources, although some aspects such as backup of data and applications may be left in the hands of the cloud service customer.

The cloud service customer is left to deal with the applications and the associated software stack, although most IaaS services do provide the operating system as part of the cloud service offering.

One way to look at IaaS is that it is a direct translation of data center capabilities into a cloud environment and so is of greatest interest to ***operators***, who are left with the decisions of how many (virtual) servers to allocate, what data storage resources to make available and what networking topologies are required.

Some assistance may be provided to application developers, where automated provision of servers, data storage and networking may be provided, principally in support of development and test processes – which can substantially shorten the lead times for these resources compared with a traditional on-premises environment.

Software as a Service (SaaS)

SaaS cloud services involve the provision of a complete application or application suite by the cloud service provider. The capabilities offered by the application can span the whole gamut of application types, including email, office productivity, customer relationship management, HR functionality and accounting.

The main characteristic of SaaS is that the cloud service provider takes responsibility for and manages the application itself and the whole software and hardware stack that the application depends on – as shown in Figure 1. Such offerings are primarily aimed at ***end-users*** of the applications – a "sign up and use" type of approach, requiring minimal investment in terms of IT operations – and no development on the part of the cloud service customer.

Many SaaS offerings are aimed at consumers, but there is an increasing number of business applications offered to enterprises as SaaS. One of the earliest and best known of such business applications is the salesforce.com CRM application.

Many SaaS offerings operate via a web browser interface to the end user, but it is also common for the user to install specialized application front-ends onto their devices to improve the usability of the application and/or take advantage of some of the capabilities of the user device, especially in the case of mobile devices such as smart phones. Such front-end applications typically operate via an application programming interface (API) to the cloud service – and it is becoming more common for SaaS offerings to make such APIs available for more general use by applications written independently of the SaaS offering itself (this is highly relevant to PaaS platforms).

It is becoming more common that SaaS offerings are built using a PaaS platform – allowing the developers of the SaaS application to concentrate on the unique capabilities of the application without having to be concerned with configuration and operation of the required software and hardware stacks, particularly where exploitation of the flexibility and scalability of cloud computing is required.

Platform as a Service (PaaS)

Platform-as-a-Service offerings principally provide an environment in which to develop, deploy and operate applications. PaaS offerings typically involve diverse application software infrastructure (middleware) capabilities including application platforms, integration platforms, business analytics platforms, event-streaming services and mobile back-end services [3]. In addition, a PaaS offering often includes a set of monitoring, management, deployment and related capabilities.

PaaS offerings are targeted primarily at ***application developers***, although PaaS offerings also typically contain capabilities that are of interest to operators.

One way of describing PaaS is that it represents a cloud service rendering of the application infrastructure offered by entities such as application servers, database management systems, integration brokers, business process management systems, rules engines and complex event processing systems. Such application infrastructure assists the application developer in writing business applications – reducing the amount of code that needs to be written at the same time as expanding the functional capabilities of the applications. The essence of a PaaS system is that the cloud service provider takes responsibility for the installation, configuration and operation of the application infrastructure, leaving only the application code itself to the cloud service customer.

PaaS offerings also often expand on the platform capabilities of middleware by offering application developers a diverse and growing set of services and APIs that provide specific functionality in a managed, continuously available fashion. This approach aims to obscure the fact that there is middleware present at all, enabling immediate productivity for developers.

In addition, PaaS offerings provide their capabilities in a way that enables the applications to take advantage of the native characteristics of cloud systems, often without the application developer having to add special code to the application itself. This provides a route to building "born on the cloud" applications without requiring specialized skills.

PaaS can be contrasted with SaaS offerings: SaaS offers a fixed set of application capabilities while PaaS supports the creation and use of application code with whatever set of capabilities is required for the business. The need for specialized code is very general – and it is telling that many SaaS offerings provide APIs specifically to provide for tailoring, customization and extension using applications built on a PaaS.

Similarly, PaaS can be contrasted with IaaS offerings: IaaS provides fundamental infrastructure but leaves installation, configuration and operation of the necessary software stacks in the hands of the cloud service customer. A PaaS offering provides the application middleware stacks ready-to-run and managed by the provider. IaaS offerings provide extensive control over resources which may be necessary for some applications, but at the cost of requiring considerable effort on the part of the cloud service customer. PaaS offerings often organize the underlying resources, removing the responsibility and effort from the cloud service customer but potentially limiting choices.

Some PaaS offerings also blend in features of IaaS and SaaS cloud services – offering some control of basic resource allocation on the one hand and providing complete off-the-shelf software capabilities on the other. This can cause some confusion – but the hallmark of a PaaS system is the ability for the cloud service customer to create and run applications and services that meet specific business needs.

Deployment Models: Public, Private, Hybrid

An important aspect of any cloud computing system is the deployment model used. There are three main deployment models that most cloud service customers should consider:

- ***Public cloud deployment*** – where the cloud service is offered publicly to many cloud service customers and the essence of the offering is that multiple customers share the resources offered by the cloud service
- ***Private cloud deployment*** – where the cloud service is used exclusively by a single cloud service customer. There are two variants: 1) on-premises, where deployment is within a data center owned and controlled by the cloud service customer organization; 2) cloud service provider, where the deployment is within a data center owned and controlled by a cloud service provider but the resources used are dedicated to one cloud service customer and are isolated from resources used by other customers
- ***Hybrid cloud deployment*** – where there is a combination of public cloud deployment and private cloud deployment, often also involving integration of the resources in the cloud services with cloud service customer assets deployed using a traditional on-premises architecture, as shown in Figure 2

PaaS systems can be deployed as public cloud services or as private cloud services. Private cloud deployment may be less efficient than public cloud deployment, due to reduced opportunities for resource sharing – increasing costs. Private cloud deployment may require the cloud service customer to have specialized in-house skills for installing and operating the PaaS system – something that can be left to the cloud service provider for public cloud deployment. On-premises deployment puts the onus on the cloud service customer to implement resiliency and backup capabilities that may be provided out-of-the-box by a public cloud deployment.

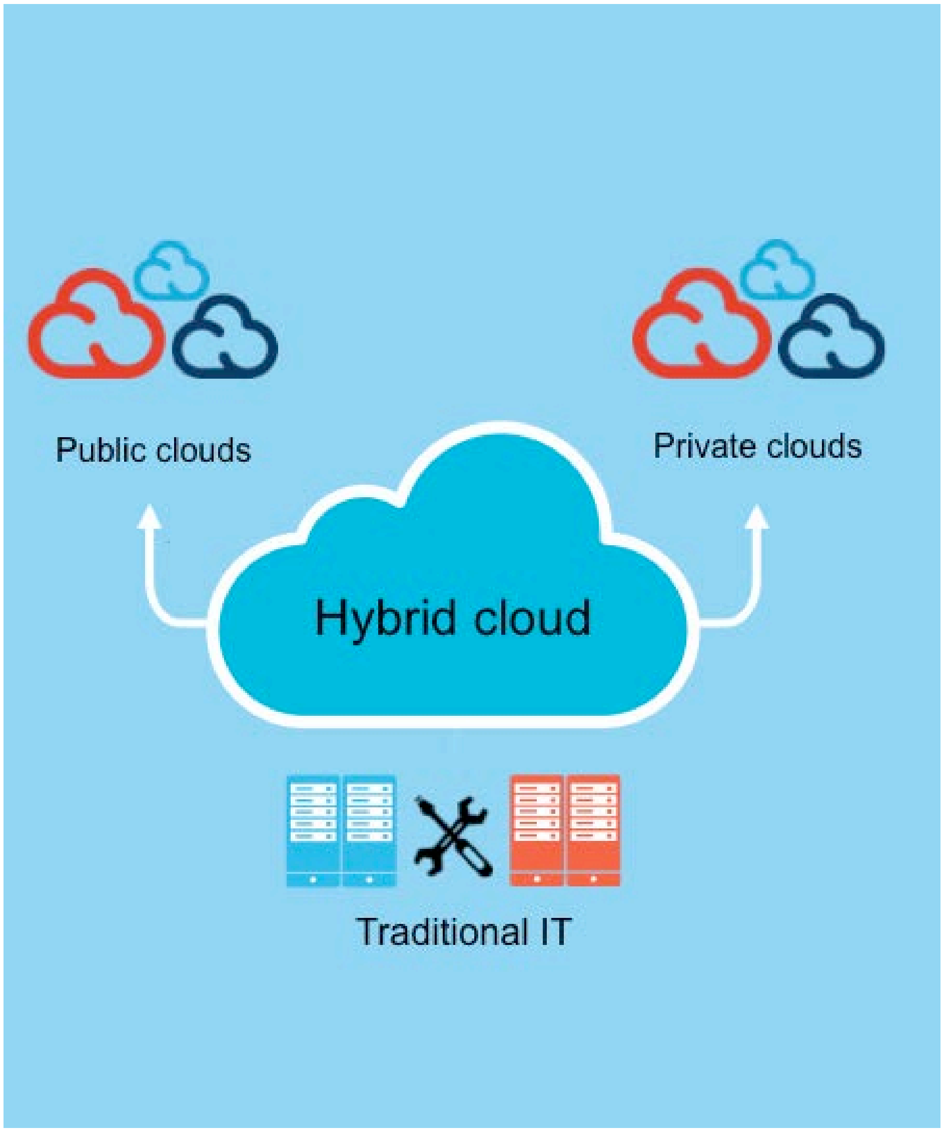

Figure 2: Hybrid Cloud Deployment

A common reason to choose private cloud deployment concerns the sensitivity and risk profile associated with particular applications, or more commonly, associated with particular datasets. Highly sensitive data, perhaps subject to regulation and legal requirements is more likely to demand private cloud deployment.

Hybrid cloud deployment may represent a sweet spot for many enterprises, with less sensitive applications and data deployed into a public cloud, more sensitive applications and data deployed into a private cloud, with integration between them providing access between the environments where necessary.

Characteristics of Platform-as-a-Service

The major characteristics of PaaS are:

- *Support for custom applications*. Support for the development, deployment and operation of custom applications. PaaS offerings typically support “born on the cloud” or “cloud native” applications that are able to take full advantage of the scalable, elastic and distributed capabilities of cloud infrastructure. In many cases, this is achieved without the application developer having to write special code to take advantage of these capabilities.
- *Provision of runtime environments*. Typically each runtime environment supports either one or a small set of programming languages and frameworks – for example Node.js, Ruby and PHP runtimes. A characteristic of many PaaS offerings is support for a range of runtime environments, rather than just one or two – in contrast to prior non-cloud middleware platforms. This enables developers to choose the most appropriate technology for the task in hand – something that is becoming more prevalent as “polyglot” environments become more accepted in the industry.
- *Rapid deployment mechanisms*. It is typical of many PaaS offerings to provide developers and operators with a “push and run” mechanism for deploying and running applications – providing dynamic allocation of resources when the application code is passed to the PaaS via an API. Configuration requirements are kept to a minimum by default, although there is the capability to control the configuration if required – for example, controlling the number of parallel running instances of an application in order to handle the anticipated workload or to meet resiliency goals.
- *Support for a range of middleware capabilities*. Applications have a variety of requirements and this is reflected in the provision of a broad range of application infrastructure (“middleware”) that supports a range of capabilities. One is database management, with both SQL and NoSQL database technologies provided. Other capabilities include integration services, business process management, business analytics services, rules engines, event processing services and mobile back-end services.
- *Provision of services*. Support for the application developer usually consists of more than just middleware – many capabilities are provided as a series of separate services, typically invoked via an API of some kind. Services are installed and run by the provider of the service, removing responsibility from the cloud service customer. For example, in the case of a database service, the responsibility for ensuring availability and reliability, for having replicas and backups of the database data, for securing the data and so on all falls on the service provider.

  Provider services are an essential concept in reducing the effort and complexity in building software systems – rather than having to install and manage some potentially complex set of software, get the capability off-the-shelf from the provider who is expert in installing, configuring and managing the necessary software.
- *Preconfigured capabilities*. Many PaaS systems are characterized by capabilities that are preconfigured by the provider, with a minimum of configuration available to developers and customer operations staff. This reduces complexity, increases productivity and lowers the potential for unexpected problems, with capabilities simpler to manage and easier to debug.
- *API Management capabilities*. It is increasingly the case that business applications need to expose some capabilities via APIs (“the API economy”). In some cases, this is required by the

nature of the user interface to the application – mobile apps usually need an API so that while operating independently of the business application, they can access data and transactions when required. In other cases, part of the business solution is to enable other parties (partners, customers, suppliers) to integrate their own business applications with those of the enterprise – this integration is done via an API. Providing an API requires a level of control, so that only authorized users can access the API and each user can only access those capabilities for which they have permission. Offering an API requires some amount of API Management.

- *Security capabilities*. Security is one of the most important aspects of any business solution and so it is not surprising to find that PaaS offerings provide built-in security capabilities, reducing the load on developers and operators to provide and manage these capabilities. Capabilities include firewall, endpoint management, secure protocol handling, access and authorization, encryption of data in motion and at rest, integrity checking, etc. PaaS systems can offer these capabilities with minimal or no impact on application code, simplifying the programmer's tasks.
- *Tools to assist developers*. An aim of many PaaS systems is to unify and streamline development and operations for systems – in other words, support DevOps by breaking down the divisions between development and operations that have commonly existed for in-house systems. PaaS systems assist the application lifecycle by providing development tools including code editors, code repositories, build tools, deployment tools, test tools and services and security tools. It is also common to find a set of application monitoring and analytics services, including capabilities such as logging, log analysis and app usage analytics and dashboards.
- *Operations capabilities*. PaaS systems assist operators through operations capabilities both for the applications and for the PaaS system itself, via dashboards and commonly via APIs that enable customers to plug in their own operations toolsets. So, for example, capabilities to increase or decrease the number of running instances of an application are common (to deal with varying application load), in some cases handled by automated services that vary the number of instances based on a set of rules established by the customer operations staff.
- *Support for porting existing applications*. While many PaaS systems are primarily designed to support "born on the cloud" applications, there is recognition that customers have existing applications that can be ported to the PaaS environment with resulting business benefits. Some PaaS systems have application environments that aim to closely match those available on existing non-cloud middleware stacks. However, the questions that hover over this type of application porting are whether the ported application will function correctly and whether the application can get the best from the PaaS environment without undergoing modifications. There is no "magic bullet" when it comes to porting an application from a non-cloud environment to a cloud service. Further advice on migrating applications to a cloud computing environment can be found in the CSCC white paper *Migrating Applications to Public Cloud Services: Roadmap for Success* [17].

PaaS systems are more commonly designed with "Systems of Engagement" applications in mind, rather than "Systems of Record" [13], [14], [15]. Systems of Engagement are those applications that are dedicated to helping end users such as customers, employees and business partners interact with the business and with each other. This is in contrast to Systems of Record, enterprise applications which are the authoritative stores of enterprise data and which provide core business processes for the enterprise. Systems of Engagement concentrate on providing useful and personalized capabilities for end users to interact and often involve social and communication features typically not found in the more traditional Systems of Record enterprise applications.

PaaS systems can be used to build applications that are then offered to other customers and users as a Software as a Service offering. The requirements of SaaS applications, including scalability and the ability to handle multiple tenants can usually be met by the cloud computing capabilities of a PaaS system.

The Benefits of Platform-as-a-Service

One of the major benefits of PaaS is that it improves application developer productivity. PaaS provides direct support for business agility – rapid development, faster more frequent delivery of functionality with business impact using continuous integration techniques and automatic application deployment.

PaaS systems enable the realization of the benefits of cloud computing as a whole:

- Scalability including rapid allocation and deallocation of resources with a pay-as-you-use model (noting that the use of individual resources can vary greatly over the lifecycle of an application)
- Reduced capital expenditure
- Reduced lead times with on-demand availability of resources
- Self-service with reduced administration costs
- Reduced skill requirements
- Support of team collaboration
- Ability to add new users quickly

The support for automation provides both productivity improvements and also consistency of delivery. Associated with automation is the ability for closer equivalence of the development, test and production environments, again improving consistency and reliability of delivery – this is one aspect of agile development.

PaaS systems typically build in security and data protection features, including resilience capabilities such as replication and backups. This can improve security and reduce the need for in-house security skills.

The provision of sophisticated off-the-shelf capabilities as services enables the rapid creation and evolution of applications that address business requirements, especially important when considering mobile and web applications that include social and internet of things capabilities.

Business applications typically require integration and involve aggregation of data and services from multiple existing systems – PaaS systems usually feature prebuilt integration and aggregation components to speed and simplify the necessary development work.

PaaS systems typically enable the sharing of resources across multiple development teams – avoiding the need for wasteful allocation of multiple assets of the same type in separated silos.

Examples of PaaS Offerings

PaaS offerings fall into two broad categories:

- PaaS cloud services, either offered as a public cloud service or as a private cloud service offering (either in a provider environment or on-premises).
- PaaS software, provided as one or more software stacks that can be used to assemble a PaaS, typically used by an enterprise to build a private cloud service offering PaaS capabilities; or alternatively, used by cloud service providers to create a PaaS cloud service.

It should be noted that there is a major distinction between a PaaS cloud service and PaaS software – if a customer chooses to install and operate PaaS software on-premises (i.e. in effect the customer runs their own PaaS service), this can be a complex undertaking requiring specialist skills and considerable effort. PaaS as a cloud service delivered by a provider puts this entire burden onto the provider and keeps things simple for the cloud service customer.

The website http://www.paasify.it/filter has an extensive listing of PaaS offerings, both cloud services and software – this site is a useful resource to appreciate the full breadth of offerings available and is kept up-to-date as new offerings arrive and existing offerings are updated.

The following table lists some of the more commonly used PaaS offerings, both as cloud services and as software.

PaaS Cloud Services	Public	Private	Open Source
IBM Bluemix https://console.ng.bluemix.net/	✓	✓	Cloud Foundry
HP Helion http://www8.hp.com/us/en/cloud/helion-overview.html	✓		Cloud Foundry
Microsoft Azure http://azure.microsoft.com	✓		
Google AppEngine https://cloud.google.com/appengine/	✓		
Amazon Elastic Beanstalk http://aws.amazon.com/elasticbeanstalk/	✓		
Pivotal Web Services https://run.pivotal.io/	✓		Cloud Foundry
Heroku https://www.heroku.com/home	✓		
Red Hat OpenShift Online https://www.openshift.com	✓		OpenShift
PaaS Software			
Cloud Foundry http://docs.cloudfoundry.org		✓	Cloud Foundry
Red Hat OpenShift http://www.openshift.org/		✓	OpenShift
Windows Azure Pack http://www.microsoft.com/en-gb/server-cloud/products/windows-azure-pack/		✓	

Governance and Business Considerations

There are a number of important business considerations that need to be taken into account when considering PaaS and specific PaaS offerings:

- Create a cross-functional team for a PaaS adoption and include staff with understanding of the key areas:
    - Internal billing and cost allocation
    - Audit and risk management
    - Information security, data protection and compliance
    - Change management
    - Development and operations

  Organizations that are at a lower maturity level regarding processes for managing software licenses, change management, information security and QA testing are likely to find that moving to PaaS could stretch their existing processes. Consider these elements within the scope of the adoption project from the beginning. The discussion to transform the processes will provide insight on other matters of security, privacy and chargeback.

- Evaluate the Cloud Service Agreement and its associated cloud SLA (for detailed guidance see the CSCC white paper *Practical Guide to Cloud Service Agreements Version 2.0* [4]). PaaS platforms can be very rich in their capabilities, but pricing, SLAs and other aspects can vary between different capabilities, especially the services provided. Cloud service customers should pay close attention to the capabilities they intend to use and assure that these capabilities meet their business goals.
- Organizations which are planning to enter the “API economy” through publishing one or more APIs based on a PaaS implementation will need to make the same effort in planning for information security and risk mitigation that would be required if the same solution were built entirely using on-premises resources.
- Examine costs and charges and whether limits can be placed on usage and expenditure. How charges are reported is also important, especially where the organization has multiple internal projects using the PaaS, to ensure that costs can be allocated against the appropriate project.
- Assess software licensing of the cloud service provider – and also of the software provider, in the case where the cloud service customer wants to install “bring your own” licensed software in the PaaS environment. Decisions on accountability for tracking and managing containers should be made upfront in the project to avoid confusion later. For example: Is the software licensed for use in PaaS? How many licenses are needed, given the scalability of the PaaS environment?

  This can get complex for typical PaaS environments that use containers to run software components where, for example, the number of cores or the amount of RAM allocated to the container is less than the whole system on which it runs.
- Consider compliance requirements – the cloud service customer may be subject to regulations and other legal and commercial requirements. The PaaS environment must be evaluated to see whether its use meets compliance needs – this may be proved by suitable certifications and audit results relating to the PaaS.

  As suggested in the section on the context for PaaS in cloud computing, considering whether the application to be ported is a ‘System of Engagement’ or a ‘System of Record’ can help business and IT teams make decisions. Breaking a complex system down into layers of functionality and data sensitivity can help guide decisions on geographic location of data storage and regional compliance matters.
- Assure that the use of the PaaS will not result in lock-in to a single cloud service provider. The ability to port applications and data from the PaaS to another PaaS, whether belonging to another provider or implemented in-house, is a key factor in avoiding lock-in. A PaaS system based on an open source project with open governance is one possible approach here. See the CSCC white paper *Interoperability and Portability for Cloud Computing: A Guide* [5] for guidance on the topic of avoiding lock-in.

Governance applies to the use of any cloud services and PaaS offerings are no exception. Cloud service customers must ensure that governance support is appropriately provided for the following areas:

- Change management, particularly with respect to software levels and patching. One benefit of PaaS is that the software stack used by applications is supplied and maintained as part of the PaaS. It can be a boon to have the software maintained by the cloud service provider. However, there needs to be clear communication about changes and appropriate forewarning of changes to the customer so that any impacts can be planned. Cloud service customers should be particularly alert for terms in the cloud service agreement that indicate that the cloud service provider can make significant changes and/or discontinue cloud services at relatively short notice (say, less than 4 weeks).
- Security, including availability, confidentiality and integrity of the applications and services in the PaaS environment. Processes for managing identities and for controlling access and authorization must be clear and there should be an appropriate level of auditing (e.g. through examination of logs). The ability to ensure that policies for encryption of sensitive data and for verifying data integrity are carried out appropriately is important.
- Processing locations and where data is stored, particularly where the PaaS resides in data centers belonging to a cloud service provider. Part of the concern is regulatory – there may be legal constraints on where certain types of processing and storage can occur (e.g. European privacy regulations). In addition, there may be enterprise policies requiring processing and data to be spread out geographically to avoid single points of failure.
- System billing– when an invoice is received, it is necessary to ensure that the charges correctly reflect the actual use of the system by the customer. For this, there must be appropriate logging of what is used and for what periods, and the ability to cross check these logs with the invoice received.
- Control of the capabilities developers and operators use within the PaaS environment. Partly this is a question of financial control, but it can also be a question of ensuring that applications only use capabilities for which there are skills and support within the enterprise (no surprises!!).
- Finally, there is a need to ensure that there is an appropriate exit process in place should the customer decide to stop using a particular PaaS offering.

Guide to Acquiring and Using PaaS Offerings

This section examines what factors should be considered when a cloud service customer seeks to buy and use a PaaS offering. It is important to understand that fully embracing PaaS implies a number of technical and organizational changes for the customer organization. While it may be possible to port an existing non-cloud based application to use a PaaS environment, this is not likely to realize major benefits. A certain degree of change is required at the technological level – and perhaps a larger degree of change is implied at the organizational level.

This section deals with the technological changes first. The organizational and process changes are discussed later, in the light of the technology – although it is important to understand that the two aspects are in fact closely linked with each other. The following areas are examined in detail:

- Understand PaaS end-to-end application architecture
- Understand how containers enable applications
- Understand how services and microservices are used
- Address integration between PaaS applications and existing systems
- Consider development tools and PaaS
- Ensure appropriate security components
- Expect support for agile development and DevOps
- Consider the deployment aspects of PaaS

Understand PaaS end-to-end Application Architecture

This section introduces the concepts contained in the "12-factor app" methodology for cloud-native applications. PaaS systems aim to support these concepts so that it becomes straightforward to write applications that can exploit the capabilities of cloud computing – it is useful to understand these concepts since they are the reason for the structure and capabilities of many PaaS systems.

The "12 factor app" concepts [6] were created by a group of developers associated with the Heroku PaaS system who identified successful patterns distilled from experiences with many applications built on Heroku. The 12 factors are summarized as follows:

1. *Codebase*. One codebase in a code repository, tracked in revision control, many deploys (e.g. to developer, staging, production).
2. *Dependencies*. Explicitly declare and isolate dependencies.
3. *Configuration*. Store application configuration in the environment, completely separated from the codebase.
4. *Backing services*. Treat services used by the application as loosely coupled attached resources.
5. *Separate build, release, run*. Strictly separate build and run stages.
6. *Processes*. Execute the app as one or more stateless, share-nothing processes.
7. *Port binding*. Export all application services via port binding.
8. *Concurrency*. Scale out via the process model – handle greater load by running multiple parallel processes all running the same code.
9. *Disposability*. Maximize robustness with fast startup and graceful shutdown.
10. *Development/production parity*. Keep development, staging, and production as similar as possible – use continuous deployment.
11. *Logs*. Treat logs as event streams – not as files.
12. *Admin processes*. Run admin/management tasks as one-off processes.

It is instructive to note that these factors are a mixture of application architecture and of development technique and process. They have a close relationship with agile software development [7] and extreme programming [8], which emphasizes an approach to application development that involves rapid and frequent deliveries of working code. An unstated aspect is that most processes become automated and test-driven: the processes are executed frequently (so demand minimum intervention) and there is a need to ensure a successful outcome in a high percentage of cases, for which good test suites are key.

One of the key architectural elements is the principle that the applications are stateless and that they use separately deployed services to build their functionality, as shown in the example in Figure 3. "Stateless" implies that the applications themselves don't retain data between requests. Data representing state is handled by one or more of the services – such as the database in the example. One reason for this approach is to support scalability – if the application needs to handle a higher level of requests from clients than can be handled by a single instance of the application, then two or more instances can be run in parallel, sharing the workload across them. This form of *horizontal scalability* is key to exploiting the capabilities of cloud computing – cloud service customers are advised to check whether any PaaS offerings they are considering can support this kind of scalability. State held within the application prevents effective sharing of workload since sequences of requests have to be routed to the application instance holding the state data.

Dividing up the application into a number of separate components ("services" and "microservices" – described in a later section) works well with this model, in that each component can be separately scaled according to the load placed on that component alone – unlike a monolithic application where the whole application is scaled up or down, usually based on the loading of the most heavily used component.

There are other advantages to this architectural approach – failover and redundancy to provide resilience and high availability is an important one, where the key ability is to have multiple instances of the application running on different hardware, possibly in widely separated datacenters.

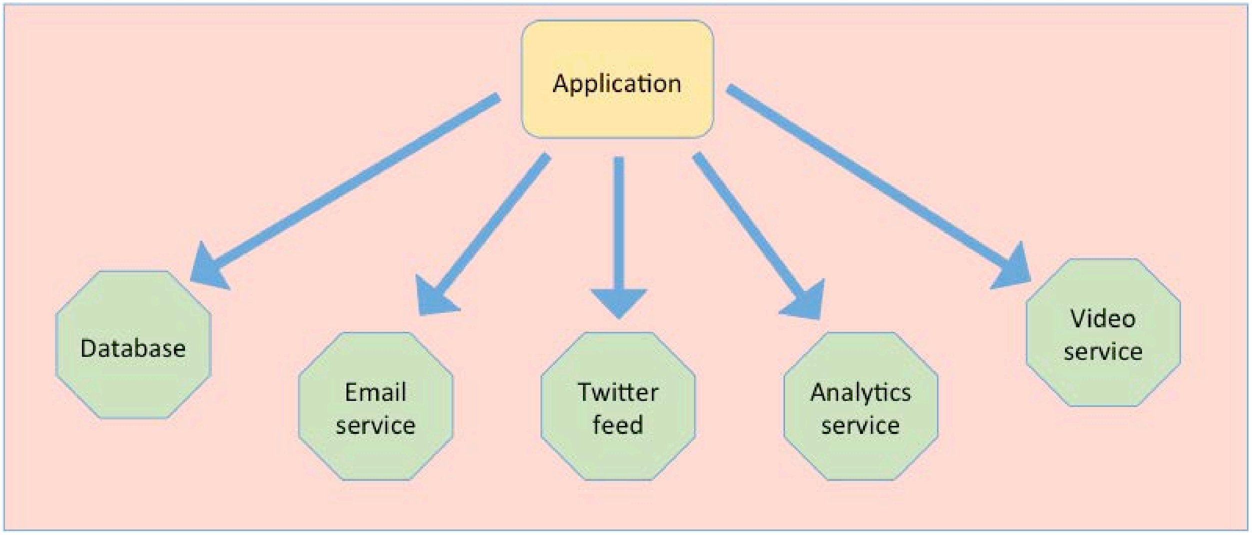

Figure 3: Typical PaaS Application Architecture

It is important to note the distinction of this form of application architecture from commonly used "monolithic" application architectures where all the components of the application are packaged, deployed and run as a single entity.

Understand how Containers enable applications

Application containers are an important aspect of PaaS systems. This section describes the characteristics of application containers and how they are used to support applications.

One of the key characteristics of cloud computing is its ability to share resources (this is fundamental to its economics, but is also important to characteristics such as scalability and resilience). Sharing of compute resources requires some level of virtualization. The original approach to software virtualization is the use of *virtual machines*, which involve a *hypervisor* providing an abstraction of the system hardware and permitting multiple virtual machines to run on a given system, each containing its own guest operating system (OS), as shown in Figure 4. This permits the system to be shared by the applications running in each VM.

VM based hypervisor virtualization provides a high level of isolation and security but suffers from a performance overhead due to the emulation of the hardware in the hypervisor. Physically, there are also multiple copies of operating systems present, which consume resources. This effectively limits the number of VMs that can share a given system. In addition, there is a performance cost in starting each new VM due to the initialization of the guest OS - in addition to whatever time it takes to initialize the application and its software stack.

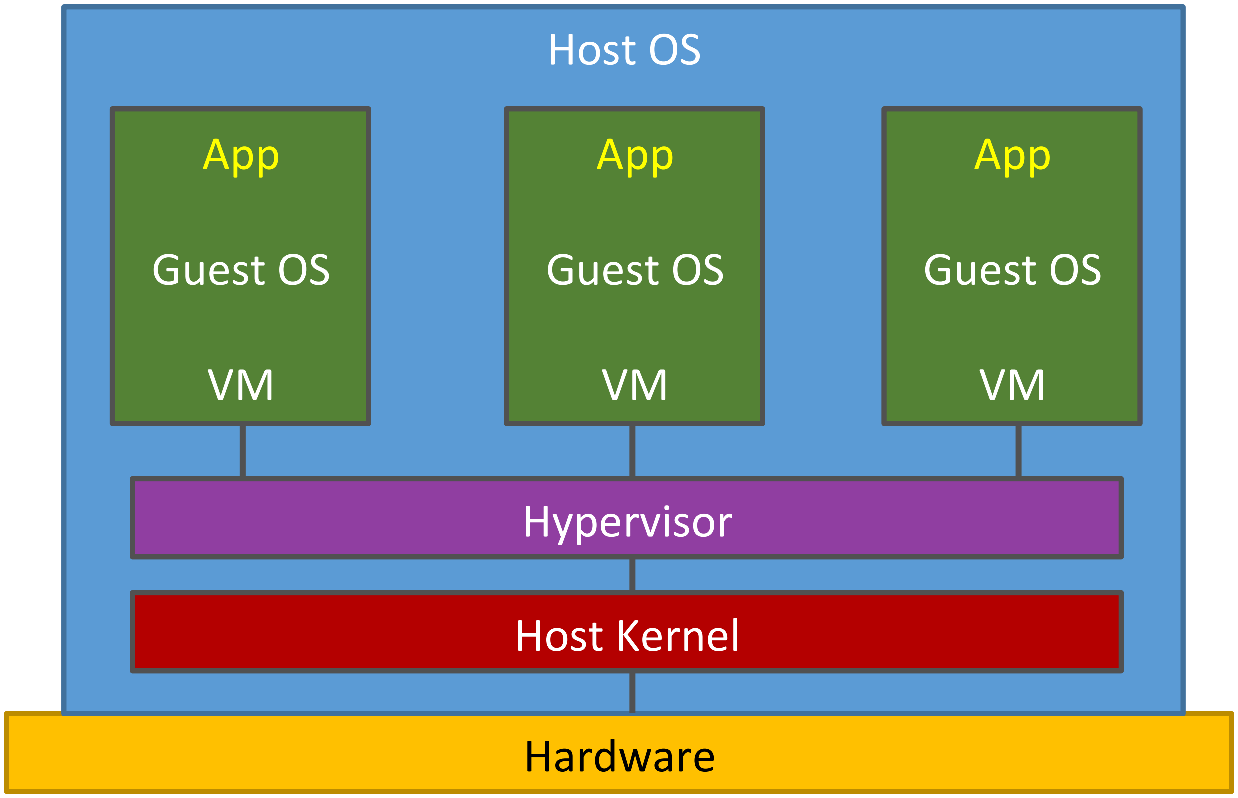

Figure 4: Hypervisor Virtualization

These drawbacks resulted in the creation of a newer, lighter weight virtualization technology – container virtualization. Containers virtualize the operating system rather than the system hardware – and all containers on a given system use the same host OS kernel, as shown in Figure 5. Each container isolates a set of processes and resources such as memory, CPU, disk – isolated from other containers and from the host OS.

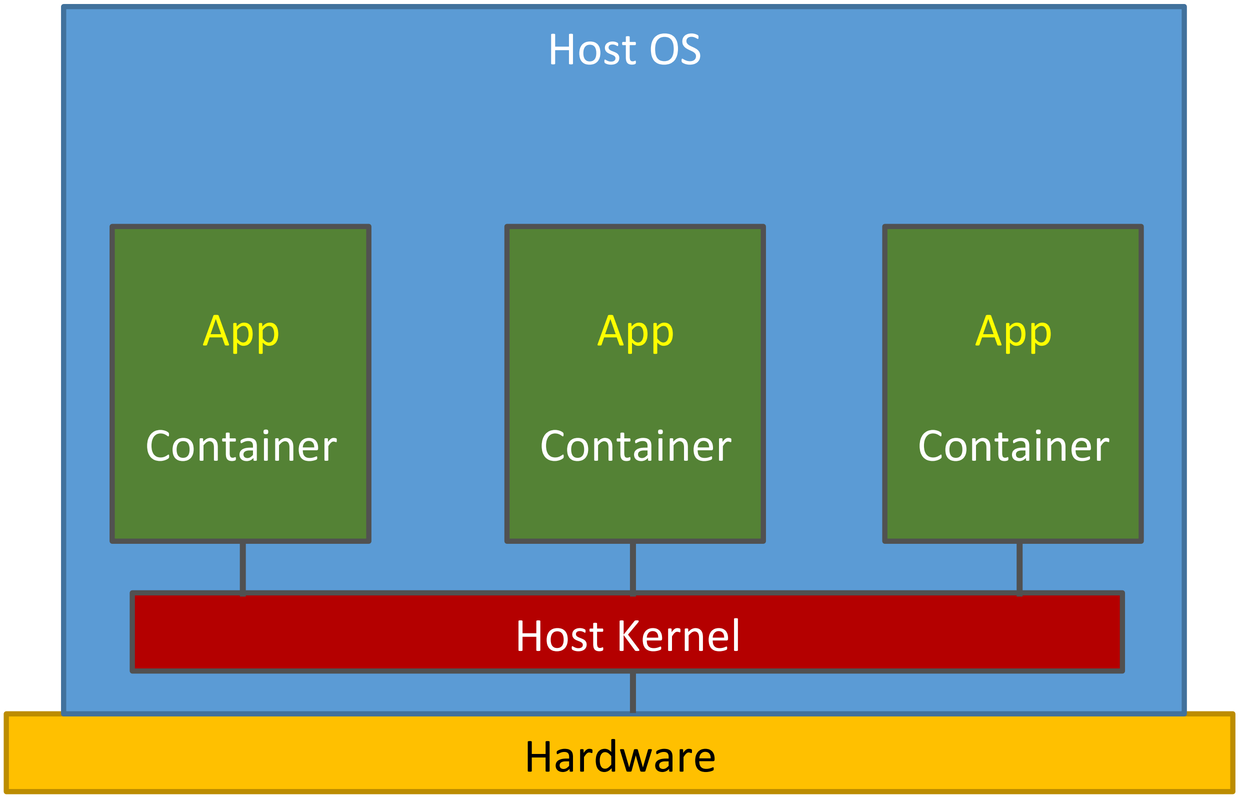

Figure 5: Container Virtualization

Containers have very low performance overhead and utilize the resources of the host system better than VMs since there is no copy of an operating system within the container, with the result that many more containers can run on a given system than VMs. Starting and stopping containers is also relatively fast, since there is no operating system to initialize – all that remains is the startup time of the application itself. There are some drawbacks to containers – first, they must all use the same operating system as the Host OS, since there is no separate copy of an operating system in the container, unlike in a VM where one OS can be hosted on a system running a different OS. Secondly, there is more concern over the isolation of applications within containers, which is not as strong as it is for VMs.

Many PaaS platforms use containers to run applications. Some, like Cloud Foundry, Openshift and Heroku have their own implementation of container technology. However, more recently, the Docker container technology has risen to prominence and has become part of the influential and widely supported Open Container Initiative [16] which appears likely to form the core of future container implementations, with some PaaS systems already committed to migrating to use the OCI runC container technology (such as Cloud Foundry). However, VMs can also be used to deploy and run applications within some PaaS systems and can have advantages for some kinds of application, particularly where there is a need to control the operating system in detail.

Each PaaS application is run within a container as a single operating system process. The container has a copy of the application code itself and also the software stack that it uses. In many cases, the software stack is a ready-prepared stack available from some kind of library system, avoiding the need for the developer to build and configure the stack from scratch. The ready-prepared software stacks have different names for the different container technologies – Heroku and Cloud Foundry call them “buildpacks”, OpenShift calls them “cartridges”. So, there are stacks for many of the popular programming language runtimes such as Java, PHP, Node.js, Ruby and so on. Most container technologies also allow programmers to create their own custom software stacks.

Applications are deployed to the PaaS as complete container stacks – combining the application code with its underlying software stack – ideally the assembly of the stack is done by the PaaS itself rather than the developer. There is usually configuration applied to the containers for the application – specifying limits for the amount of RAM that can be used, for example. The PaaS system is then able to start one or more containers to run the application on one or more of the systems available to the PaaS.

The lightweight nature of the application containers means it is quick for the PaaS to spin up a new running instance of the application – and to run multiple copies in parallel in separate containers. Where multiple instances of an application are running, it is common for the PaaS to arrange to distribute incoming requests across all of the instances using some form of load balancer. This model also supports the principle of dividing the application into a series of separate components, each separately deployed and separately scaled, as described in the previous section.

Container technologies are evolving rapidly and it is important for the cloud service customer to ensure that the container technology of a PaaS is going to be able to adapt to these rapid changes, with minimal impact on the customer applications. Ideally, the container technology is virtually invisible to the application code, but this may not always be the case. A strong ecosystem supporting the container technology is perhaps the best indication of an ability to deal with changes and evolution of the technology.

Understand how Services and Microservices are used

Services and microservices simplify the development and the operation of applications in a PaaS system.

Services can reduce the amount of application code that has to be written and typically offer standard ready-to-use capabilities with known operational characteristics. Services can be provided as part of the PaaS offering or may be provided by third parties. It is common for a PaaS offering to have a catalog of available services that can be used by developers. It is typical for services to be accessed via an API – the “API economy” that puts needed capabilities at the fingertips of developers, freeing them to deliver innovative solutions to the enterprise.

Microservices describes a design approach for building the application itself, dividing the application into a series of separate processes, deployed independently and connected via service interfaces. Microservices architecture is described in detail in [9] and [10]. The concept is that the microservices within the application should be designed to implement some specific area of function, perhaps a particular business process. It should be possible to operate and to update each microservice independently.

One of the advantages of using a microservices architecture is that each component of the application can be scaled separately to match the load on that component alone. This differs from the typical “monolithic application” approach used in older enterprise applications – where everything is deployed and operated as a single entity, with scaling only possible by scaling up/down the whole application, leading to resource inefficiency for those parts of the application not under heavy load. PaaS systems make it straightforward to deploy each microservice independently and link them together to create the full application. Each microservice can be managed independently – scaled, distributed, updated.

Developing an application using microservices architecture is complementary with using sets of services for specific capabilities within the application. For services, PaaS systems make service instances available for immediate use with minimal effort.

An example of a service is a database management system. A DBMS can be a complex beast to set up and operate, particularly considering aspects such as replication (for resiliency) and backups. Rather than including the DBMS as a component of the application, it is becoming more common for the DBMS to be installed and run independently of any particular application. Many DBMS systems are provided as cloud services, such as MongoDB, Cloudant and ElephantSQL. A DBMS service instance is allocated for a particular application and accessed through a service interface. The service provider is responsible for the availability of the service – and also for critical factors such as scaling the service to deal with variations in workload.

One useful characteristic of using services to build applications is the use of different instances of the same service for development and production. Developers can use a development instance of a database, while the production deployment of the same application can use a production instance of the same database. This approach helps address Factor 10 of the 12-factor app.

It is important for cloud service customers to understand how a PaaS offering provides support for microservices and services – and also what set of services are supplied as part of the PaaS (the catalog).

Address Integration between PaaS Applications and existing Systems

Seldom are applications developed and run on a PaaS platform completely standalone – commonly these applications must integrate with existing enterprise systems – applications, services, data – some of which are likely to reside in an existing enterprise environment outside the cloud system.

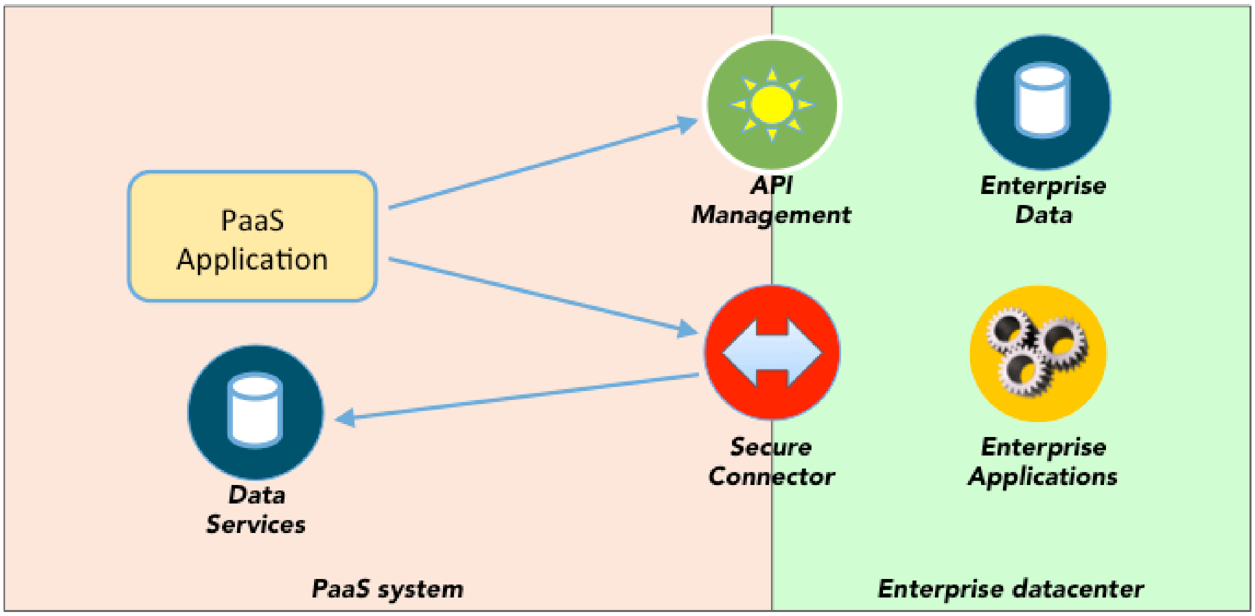

Figure 6: Integrating PaaS Applications with Existing Systems

A variety of approaches and technologies are available to connect the PaaS application to the existing enterprise systems.

An API Management component can be installed with the purpose of exposing a controlled set of capabilities of one or more of the enterprise systems as an API that can be called by the PaaS application. The API exposed can be a transformed form of the enterprise applications and data – data formats can be mapped and the API service operations can be aggregations or restrictions of enterprise functions. The API Management component also manages the security of the API, typically limiting API access to one or more specific PaaS applications.

An alternative approach is a Secure Connector component, which can handle application-to-application connectivity and can also provide database-to-database connections. The database connectivity approach is useful where direct access from the PaaS application to the enterprise data is undesirable, for example where the number of transactions from the PaaS application is too great for the enterprise data system or where the data must undergo a complex transformation to be usable by the PaaS application (common where the PaaS application is supporting an interactive web or mobile application). The database-to-database connector provides the capability to move and transform data between enterprise data systems and one or more data services in the PaaS environment, with the PaaS data services being designed to handle the interactions with the PaaS application.

Differing approaches are possible for mapping data between the PaaS system and the enterprise data systems – the connection can be done on a batch-style basis, or the enterprise data can be updated whenever the PaaS application does a transaction which changes the enterprise data. The choice is made based on the nature of the application and the capabilities and capacities of the enterprise systems.

Cloud service customers must examine their existing enterprise systems and understand how those systems can be linked to applications and services running in the PaaS system.

Ensure appropriate Security components

Security and data protection for personal data are key elements of any information system and so it is important that the PaaS offering provides appropriate capabilities to enable end-to-end security for deployed applications.

Cloud service customers should consider security and data protection capabilities of two types – firstly the tools and facilities that help developers build secure applications and services, secondly the services and capabilities provided by the PaaS systems to secure and protect applications and services.

Tools and facilities to assist developers include capabilities such as application scanning tools to check for endpoint vulnerabilities; for example, open ports; authentication for use of open ports. Static analysis tools can also provide help in exposing weaknesses in applications. Load impact tools are also valuable in checking the behavior of the application under stress and help assure the availability of the application. There is also the question of the protection of the application code itself – there needs to be appropriate authorization and tracking in the code repository and integrity checking of the deployed code.

Facilities to support security and data protection for applications themselves include:

- Firewalls, which are commonly offered as part of the PaaS system. Often associated with firewalls is DDoS attack handling.
- Secure communications handling (e.g. HTTPS), which may also include the handling of certificates, since it is highly likely that cloud service customers want to use their own domain endpoint addresses and their own security certificates.
- API management – handling of any exposed APIs, including control of which interfaces are offered to which users, and also the management of authentication and authorization.
- Infrastructure and network protection capabilities, enabling monitoring and alerts for the communications between application components and also with underlying services.
- Identity and Access Management capabilities, including support for capabilities such as single sign-on, federation of identity management (typically with cloud service customer IdAM systems) and the use of one of a range of authentication systems (OAuth2, SAML, etc.). Multi factor authentication and the use of biometrics are growing in importance – as also is the ability to utilize mobile device technologies as part of the authentication process.
  Granular access management is a useful feature since there are usually different classes of application users who need different access rights to different capabilities of the application.
- Encryption of data at rest and data in motion is vital – the associated management of encryption keys is also highly important.
- Data activity monitoring, tracking queries and updates to stored data and in particular any activity involving sensitive data. This can include data provenance tracing to establish where each piece of data originated.

- Integrity checking capabilities, to check that data has not been tampered with.
- Resilience capabilities, including replication/redundancy of data and applications, backup capabilities and high availability techniques.
- Data management capabilities including data location control and secure deletion.
- Security information event management, including analytics of events and log data to reveal activity with security implications and to help with remedial actions.

Cloud service customers are advised to check for these capabilities in a PaaS system and assure themselves that they meet their security and data protection requirements.

Consider Development Tools and PaaS

PaaS systems typically support the development of applications that then run on the PaaS system. This means that there is a need for a variety of tooling: code editing, code repositories, build functions and test capabilities. Such tools can either be provided as part of a PaaS offering, or alternatively the PaaS offering can provide open interfaces that enable third-party tools to be plugged in by the developers.

Some PaaS systems include cloud-based tooling, where all tools run in the cloud environment, typically accessed via web browser interfaces. An example of cloud-based tooling is the Eclipse Orion package [12] – which provides facilities like browser based code editors. It is very common to provide a code repository as part of the cloud service.

Developers may prefer to use workstation based tools to develop code – this can include simple command line tools, code editors, sophisticated development environments such as Eclipse, plus test tools. Cloud service customers should understand the tooling that their developers will use and how these tools integrate with the PaaS system.

One of the main concerns for workstation-based tools is to ensure that the developer has appropriate access to the environment that the application code has when it is running in the PaaS system, in order to test and debug code.

One approach is to provide local mockups of the services and APIs that will be used in the PaaS system. Another approach is to enable remote access to (development) versions of the actual services in the PaaS system. A more sophisticated approach that is now becoming available for some PaaS systems is to support local test and debugging tools operating against the code running in the PaaS system itself (“remote debug”).

The drawback with local mockups is that they can take considerable effort to create and may be difficult to keep in step with any changes to the real services they mimic – however, mockups are a normal part of test driven development, so that they are not unusual for developers to deal with.

Remote access to the PaaS services can work well – since the functionality is “the real thing”, although there may be some concerns over security which need to be addressed – for example, it is typical for any services exposed in this way to be development instances of the services, completely separated from the production services.

Remote debug capabilities are an interesting development that may put developing PaaS applications onto the same footing as developing for on-premises systems – cloud service customers are advised to ask their PaaS providers whether such capabilities are available.

Application developers should have access to tools that enable them to control activities in the PaaS – for example, uploading ("pushing") application code, binding services to applications, controlling application configuration, starting and stopping application instances. Such capabilities should be provided in a way that fits well with the other tools used by the developer – command line tools, graphical tools, embedded components for development environments. Ideally these tools should work via an API that is exposed by the PaaS system – cloud service customers should look for these APIs and assure themselves that the API can be used by a variety of custom tooling code.

Expect support for Agile Development and DevOps

One of the major goals of PaaS systems is to speed the development of applications – and their deployment into production and subsequent operation. This requires capabilities that cover the whole lifecycle of an application – capabilities that are automated.

A PaaS system should offer good support for agile development techniques. One aspect of the agile approach is the concept of continuous integration – an emphasis on getting a working application quickly and incrementally adding functionality frequently with rapid deployment to production. This can allow for early user testing of new functionality and rapid feedback to the developers.

There are a number of features of a PaaS system that should support agile development:

- Agile implies test-driven development, where there are tests for each piece of application functionality. This is assisted by the existence of test capabilities in the PaaS, such as test tools and frameworks.
- PaaS systems can help testing by providing for rapid allocation and deallocation of resources required for tests to run. It is also ideal for the test environment to closely match the production environment – made relatively easy where both environments are based on a single PaaS system.
- Automation of the build, test, QA and deployment processes is a key – and most PaaS systems can support this, allowing common automation tools to drive these processes, using PaaS APIs to control resources, all using a seamless environment. Sometimes the automated build tools are built in to the PaaS system, but it can be equally important for the PaaS to support the use of popular tools such as Jenkins [11].
- "Blue/green deployment" is a technique often used to introduce a new version of an application into production, implying parallel operation of old and new versions of the application, with gradual cutover from old to new. PaaS systems offer support for this approach.
- Another key element for PaaS systems is the provision of ready-to-use services that enable developers to produce the required working functionality more quickly by writing and testing less code – for example, a single-sign-on service that handles authentication and authorization.

- PaaS systems typically provide operations capabilities that address deployment, scalability of applications and services, the placement of runtime instances and management of logs. Many of these operations capabilities are automated via policy and operations services in PaaS systems, reducing the need for active intervention by operations staff.
- Requirements of operations staff for visibility of the behavior of applications and services and the generation of alerts when incidents occur or operations go outside preset boundaries are typically addressed in PaaS systems, with both the provision of monitoring services and dashboards, plus APIs through which monitoring and management capabilities can be handled by custom operations applications.

PaaS support for these aspects of agile development and DevOps should be investigated by cloud service customers.

Consider the Deployment Aspects of PaaS

As with other cloud services, there are alternative deployment options to consider for the PaaS platform – private cloud, public cloud or hybrid cloud. The basic concepts relating to these alternative forms of deployment are described in the section “Deployment Models: Public, Private, Hybrid.”

Choosing a private deployment of a PaaS system on-premises may give the cloud service customer greater control over the PaaS environment and potentially better security, but may result in a smaller catalog of available services for applications. However, the task of installing and operating a PaaS platform along with a suitable range of services should not be underestimated, especially to reach production quality service level objectives for availability, performance, security and privacy. In addition, the cloud service customer takes on more responsibility, including responsibility for the underlying Infrastructure-as-a-Service, negating many of the benefits of adopting PaaS.

A recommended approach is to examine the elements of the application solution(s) proposed and to ask where each element can be deployed to best advantage. Broadly, three elements need to be considered:

- the application itself (including microservices)
- the sets of services used by the application
- the datasets used by the application

It may be the case that one or more of the datasets used by the application are sensitive – it may be best to keep such datasets on-premises, although deployment in secure form into a public cloud environment may be a valid approach. Less sensitive datasets can be deployed in a public cloud.

It is possible to deploy the PaaS applications in a public cloud environment, even where the applications use on-premises data since it is possible to provide secure, controlled access to the on-premises data for those applications. However, it may be the case that the risk profile applying to those applications means that they must be deployed into a private PaaS environment.

It is likely to be the case that many services useful to the application exist in the public cloud environment – and some of those services may be difficult or expensive to deploy on-premises. Even if an on-premises private deployment is chosen for the PaaS applications, it may still be best for the application to connect to public cloud services.

In many cases, it is best to consider a hybrid deployment of applications, services and data, where each element is placed in a private cloud environment or in a public cloud environment or left "as-is" in an existing on-premises environment, based on the risk profile and costs associated with each element. There is no one right answer.

One aspect of hybrid deployment to consider carefully is the need to manage both the private and the public PaaS systems – it may be advantageous to use the same technology base for both deployments, guaranteeing both interoperability of the systems themselves and portability of artifacts between the systems.

Summary of Keys to Success with PaaS

Key Factor	Description
1. Perform cost/benefit analysis	● Consider the costs of developing new applications and systems using the PaaS. Consider the costs of moving existing applications, services and data to the PaaS. How do these costs compare with continuing to use in-house facilities and systems? ● Consider speed of development and time-to-market for any new or changed capabilities. Shorter development times imply reduced costs and also better ability to exploit market opportunities.
2. Assess your security risk versus innovation imperative	● Speed to market can also be speed to risk unless your information security plan is thorough and comprehensive enough to deal with the additional risks posed by the use of cloud services. ● Ensure that the security of the PaaS environment matches that of equivalent existing applications and systems.
3. Assess value provided by a given PaaS versus alternatives	● Examine the capabilities provided by the PaaS. What application technologies does it support (languages, frameworks)? What runtime configurations are supported? What services are available for developers to use? What developer tools are supported? Does it have an open pipeline? ● What is the cost model of the PaaS – is it clear how costs will scale up as the use of applications and services grow? How does this compare with other offerings?
4. Consider the long term viability of PaaS provider ecosystem	● Is the PaaS offering proprietary to a single vendor, or is it a platform that is offered by a number of competing vendors? Does it have a large vibrant ecosystem of offerings, services and skilled engineers? ● When multiple service providers offer the same platforms the financial stability of the provider and their history of service becomes a key differentiator. ● Does the provider offer a wide variety of production-ready services and the ability to integrate third party offerings?
5. Consider portability both in terms of lock in and deployment model (public, private, hybrid) as needs evolve.	● Can an application developed and deployed on the PaaS be easily moved to another PaaS, or is the PaaS proprietary and exhibits vendor lock-in characteristics? ● How strong is the technical and financial base for the PaaS: viability of the platform and its long-term technology roadmap are as important for PaaS as for on-premises software. ● Can the PaaS be provided in a variety of deployment models – public, private hosted, private on-premises and hybrid combinations of these? Can applications and services be migrated between these different deployment models?

6. Ensure integration between PaaS environment and on-premises systems is straightforward and secure.	● New or migrated applications running in the PaaS environment typically require integration with existing applications, services or data that are on in-house systems. Integration must be possible, in a simple and secure fashion. ● Consider also integration from on-premises applications and services to applications and services in the PaaS environment, particularly where there is migration of in-house applications and data to the PaaS.
7. Consider transforming to Agile methods and DevOps processes	● To get the very best out of PaaS systems, plan to move development and operations teams to use agile methodologies and organizational structures that break down the divisions between development and operations.

Works Cited

[1] ISO/IEC (2014): *Cloud Computing Reference Architecture.*
http://standards.iso.org/ittf/PubliclyAvailableStandards/c060545_ISO_IEC_17789_2014.zip

[2] ISO/IEC (2014): *Cloud Computing Overview and Vocabulary.*
http://standards.iso.org/ittf/PubliclyAvailableStandards/c060544_ISO_IEC_17788_2014.zip

[3] Gartner (2014): *Platform as a Service: Definition, Taxonomy and Vendor Landscape, 2014.*
https://www.gartner.com/doc/2833022/platform-service-definition-taxonomy-vendor

[4] CSCC (2014): *Practical Guide to Cloud Service Agreements Version 2.0, 2014.*
http://www.cloud-council.org/CSCC_Practical_Guide_to_Cloud_Service_Agreements_Version_2.0.pdf

[5] CSCC (2014): *Interoperability and Portability for Cloud Computing: A Guide, 2014.*
http://www.cloud-council.org/CSCC-Cloud-Interoperability-and-Portability.pdf

[6] 12factor.net (2014): *The 12 Factor App.*
http://12factor.net/

[7] Agile developers (2001): *Manifesto for Agile Software Development.*
http://agilemanifesto.org/

[8] Down Wells (extremeprogramming.org) (2013): *Extreme Programming – a gentle introduction.*
http://www.extremeprogramming.org/

[9] Martin Fowler & James Lewis (2014): *Microservices.*
http://martinfowler.com/articles/microservices.html

[10] microservices.io (2014): *Microservice architecture patterns and best practices.*
http://microservices.io/

[11] Jenkins CI community (2015): *Jenkins Continuous Integration Server.*
https://jenkins-ci.org/

[12] Eclipse community (2015): *Eclipse Orion development environment.* https://orionhub.org

[13] IBM jStart (2014): *Systems of Engagement & the Enterprise.* http://www-01.ibm.com/software/ebusiness/jstart/systemsofengagement/

[14] Josh Bersin, Forbes (2012): *The Move from Systems of Record to Systems of Engagement.* http://www.forbes.com/sites/joshbersin/2012/08/16/the-move-from-systems-of-record-to-systems-of-engagement/

[15] AIIM.org (2010): *Systems of Engagement and the Future of Enterprise.* http://www.aiim.org/~/media/Files/AIIM%20White%20Papers/Systems-of-Engagement-Future-of-Enterprise-IT.ashx

[16] Open Container Initiative (OCI) (2015): *Open Container Initiative.* https://www.opencontainers.org/

[17] CSCC (2013): *Migrating Applications to Public Cloud Services: Roadmap for Success.* http://www.cloud-council.org/Migrating-Apps-to-the-Cloud-Final.pdf

Appendix A: Acronyms & Abbreviations

Abbreviation	Meaning
CSCC	Cloud Standards Customer Council
IaaS	Infrastructure as a Service
IdAM	Identity and Access Management
ISO	International Standards Organization
PaaS	Platform as a Service
PII	Personally identifiable information
SaaS	Software as a Service