Platform-as-a-Service

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 .

- 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.

- The ISO/IEC cloud computing standards 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.

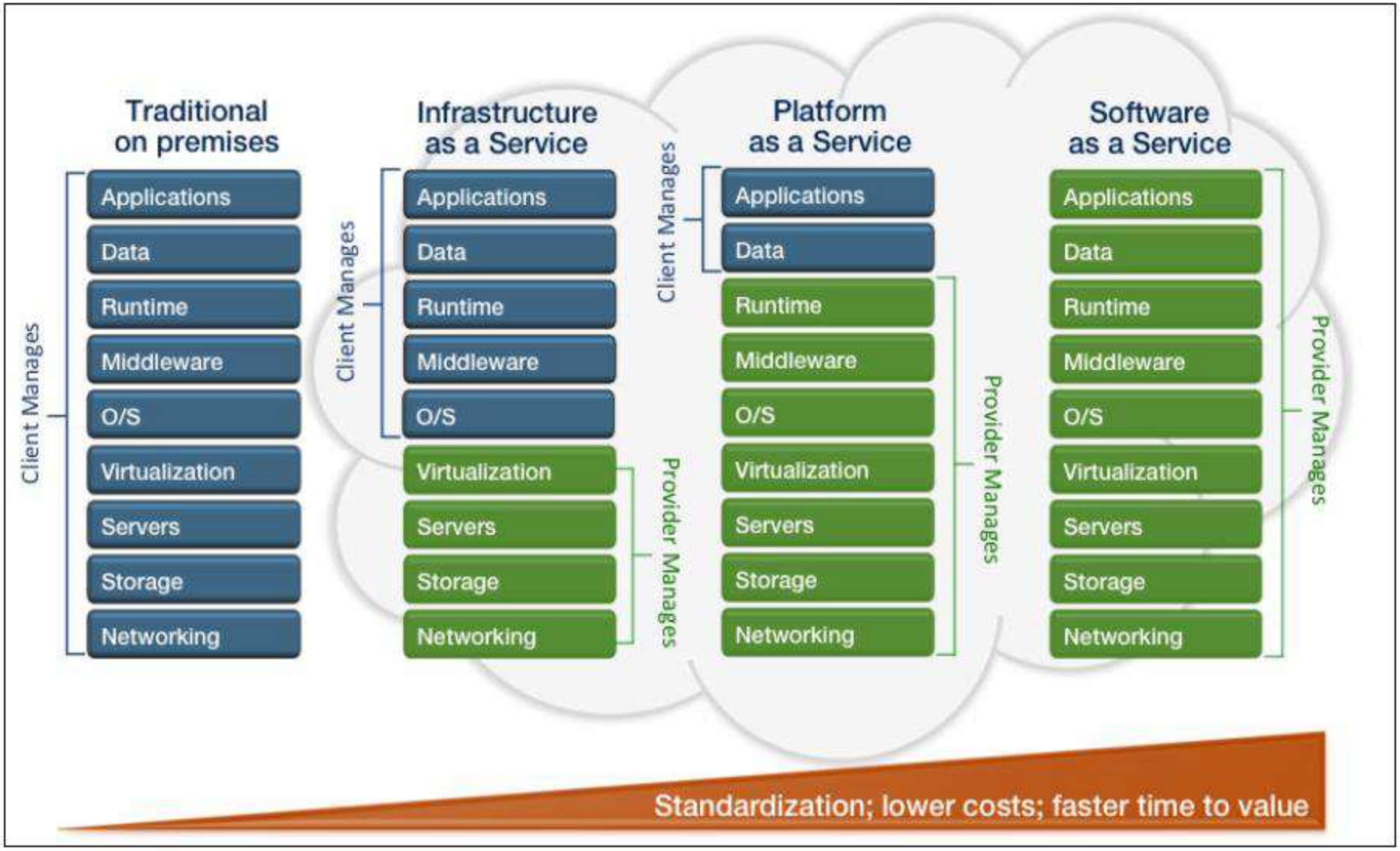

Figure 1: Application deployment - traditional and cloud-based

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

Deployment Models: Public, Private, Hybrid

- **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,

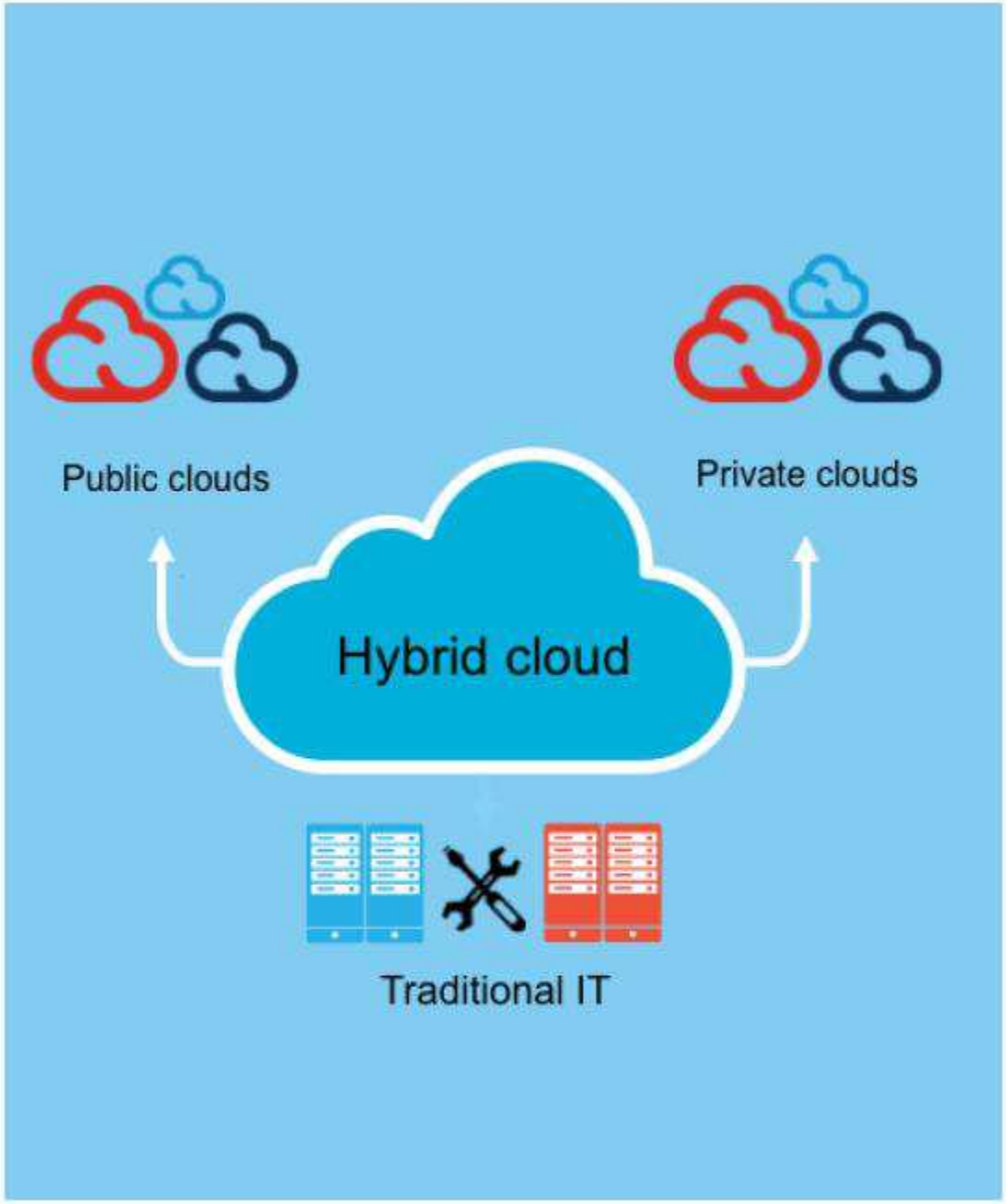

Figure 2: Hybrid cloud deployment

Characteristics of Platform-as-a-Service

- ***Support for custom applications.*** *Support for the development, deployment and operation of custom applications.*
- ***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.*
- ***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*

- **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.*
- **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.*

- **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.*
- **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.*
- **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.*
- **Tools to assist developers.** *An aim of many PaaS systems is to unify and streamline development and operations for systems*

- **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*
- **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.*
- **PaaS systems have application environments** that aim to closely match those available on existing non-cloud middleware stacks.

The Benefits of Platform-as-a-Service

- 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

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.

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/		✔	

Guide to Acquiring and Using PaaS Offerings

- 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

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.*

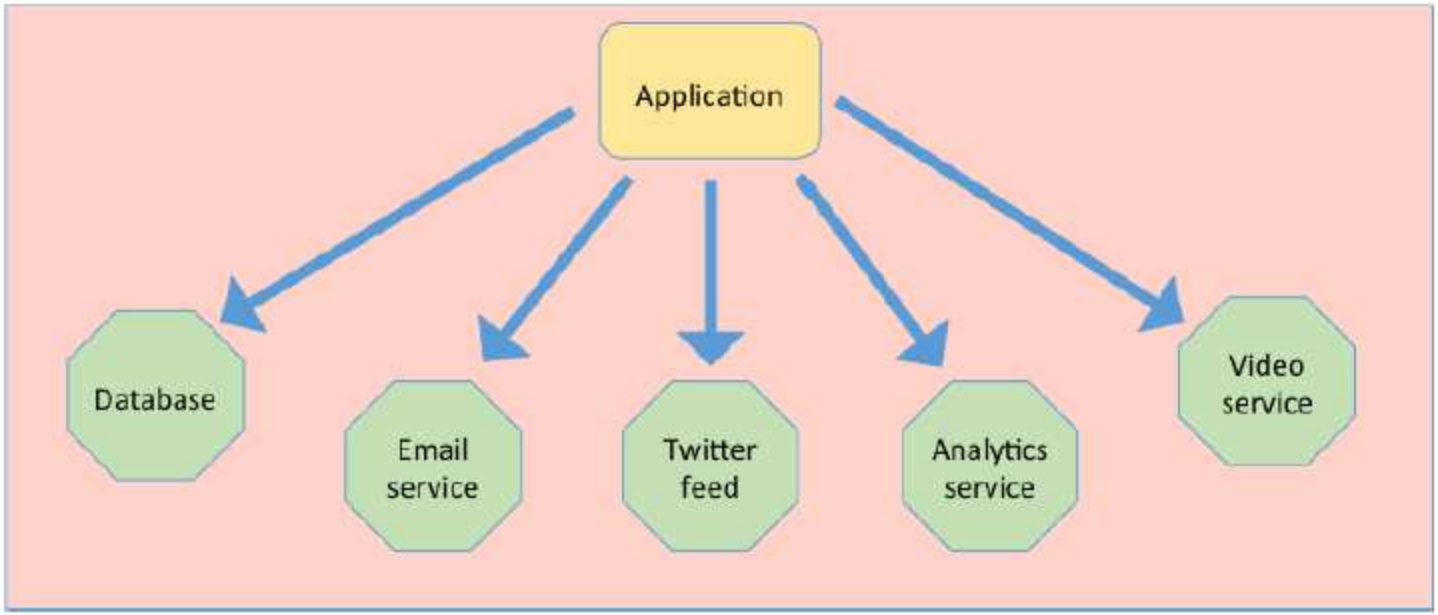

Figure 3: Typical PaaS application architecture

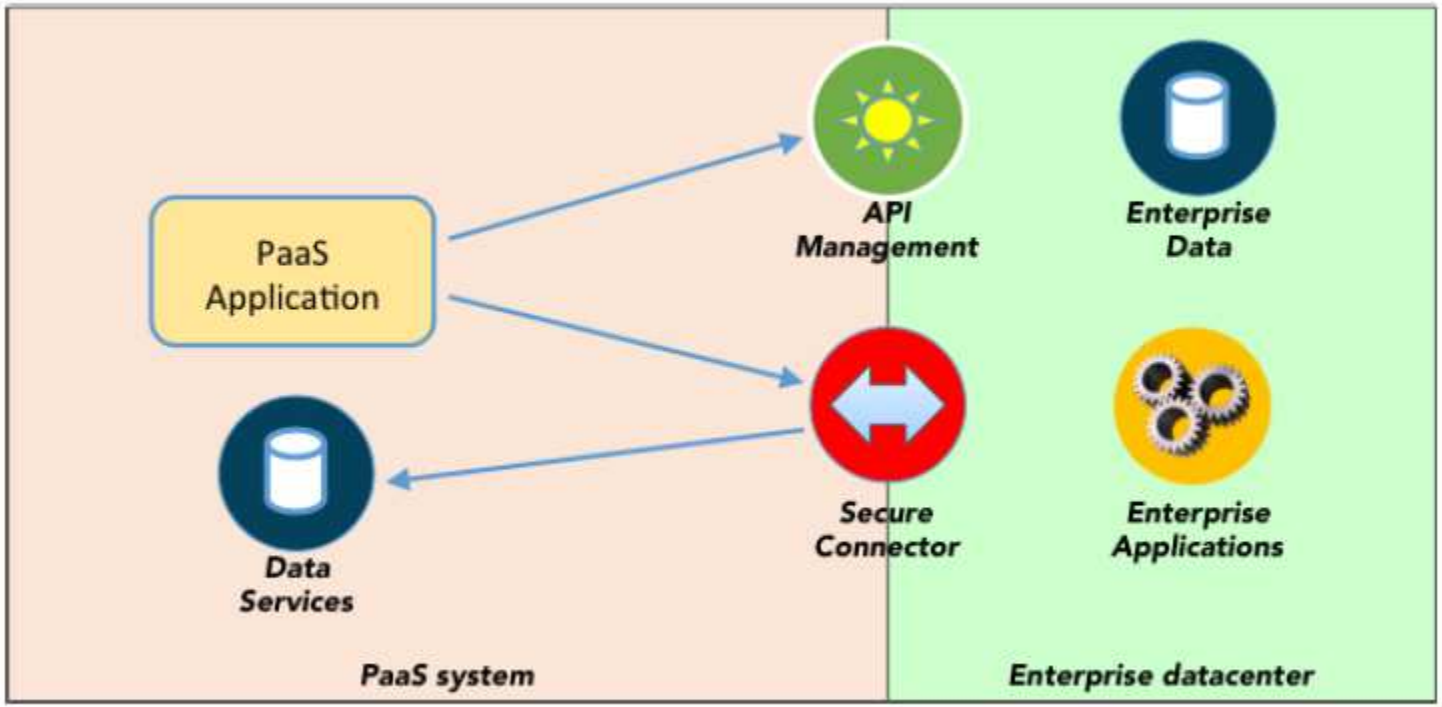

Figure 6: Integrating PaaS applications with existing Systems

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.