Private PaaS

The Next Era of Enterprise IT

apprenda®

Private PaaS: The Next Era of Enterprise IT

Executive Summary

Public PaaS, for a variety of reasons, is not accessible to a majority of enterprise IT use cases. PaaS, however, provides significant value ranging from the automation of typically mundane and long running tasks such as application deployment, to providing a foundational architecture for guest application scalability. The ability to deploy a PaaS within your enterprise IT infrastructure, a "private PaaS", allows for your enterprise developers to access PaaS value without the accessibility problems of public PaaS. By deploying a private PaaS and moving current and future custom applications to the private PaaS, your enterprise IT organization will experience significant value including:

- **Faster time to market:** Private PaaS provides for a self service utility model that allows you to upload your compiled code and in a button click, "publish" it. Never configure an application or server again. Deploy apps in minutes rather than weeks or months.
- **Increased Agility:** Leveraging private PaaS simplifies application deployment and management, and increases developer productivity through shared services.
- **Reduced Costs:** Private PaaS allows for greatly improved infrastructure utilization, removes human configuration tasks where appropriate, and provides self-service interfaces.
- **Reduced Complexity:** Private PaaS simplifies ongoing application management by abstracting applications away from infrastructure and enforcing a common, inheritable architecture.
- **Streamlined Application Management:** Private PaaS enables you to manage all of your applications from a central place and never worry about being outside the bounds of IT governance.

The prospect of leverage cloud architectures like PaaS within the enterprise IT infrastructure provides a "best of both worlds" outcome: significant cloud-based improvements in the enterprise IT experience without the adoption hurdles associated with public PaaS.

Introduction

The sea change of the past decade in enterprise IT has been driven by SOA and virtualization. These technologies have brought tremendous value to numerous organizations, solving problems related to agility and complexity. SOA has fulfilled the promise of a proper, reusable architecture pattern, and virtualization has delivered a clean, flexible approach to managing unwieldy infrastructure requirements. As with any technology, however, neither has provided a panacea to all that ails enterprise IT strategies. Significant issues still exist, ranging from how custom applications are architected and written by enterprise software developers all the way through cumbersome deployment and management processes that can typically delay releasing a new custom application by 30-45 days. Enterprise IT has not realized the vision of rapid development, deployment, and management of new applications, nor has it pushed the envelope with respect to driving cost down and efficiency up.

The era of cloud computing, and in particular of platform as a service (PaaS), which will be described a bit later, has been touted as the newest in technology shifts that promise to revolutionize enterprise IT. Cloud computing will undoubtedly drive agility and cost savings to new heights. Today, enterprise software developers can write applications using traditional programming languages and modern architecture patterns, and deploy those applications in the cloud to a public PaaS. In addition to providing commoditized platform services (e.g. caching), the PaaS model allows enterprise software developers to bypass internal infrastructure and to avoid becoming entangled in the cumbersome internal procedures required to deploy and manage their newly developed app.

While the promise of public PaaS is appealing, for a variety of reasons ranging from security to performance, a majority of enterprise use cases cannot leverage public PaaS. This poses a problem. A PaaS, as an operational layer in a computing infrastructure, provides tremendous value–with or without the "outsourced IT" provided by a public option–but is typically tightly coupled to a public deployment model where a third party both built and operates the PaaS. This tight coupling is awkward and counterproductive. Assuming that a PaaS software layer must be coupled to a public context in order to provide value is a fallacy that we readily dismiss in analogous scenarios.

For example, the non-motorized bus (as in the road vehicle that carries multiple passengers) was first put into use by Blaise Pascal in 1662 to support a public mass transit line. Although the bus was first deployed to support public transit by Pascal, the utility and value of the bus, namely the ability to amortize transport costs across many individuals, goes well beyond public transit. Modern motorized buses are used by private agencies including private charter companies, police departments, tourism agencies, sports teams and many more. To state that PaaS is valuable only in its public form factor is akin to stating the bus is valuable only in the context of public mass transit: clearly a broken world view. The analogy is not attempting to discredit public mass transit, but only to highlight the tremendous value of the bus in general.

This brings us to the vital question for enterprises: how can enterprise IT secure the value of PaaS without the adoption friction inherent in public offerings?

This whitepaper explores the advantages of deploying a PaaS tier on private infrastructure, creating a 'private PaaS' that provides an enterprise and its application developers the full benefits of public PaaS without the outsourced infrastructure hosting. Specifically, this whitepaper will describe the following:

1. The concept of a private PaaS
2. Why the software layer that enables PaaS and not the outsourcing function provided by the public form factor is the primary value driver
3. Advantages of deploying a private PaaS
4. How Apprenda is designed as a "plug and play" private PaaS for custom Microsoft .NET web and SOA applications.

The whitepaper does not intend to prove that private cloud or public cloud is better. Rather, it illustrates that the software technologies to enable public cloud, and specifically public PaaS, are extremely useful in the private context, and provide significant ROI on their own.

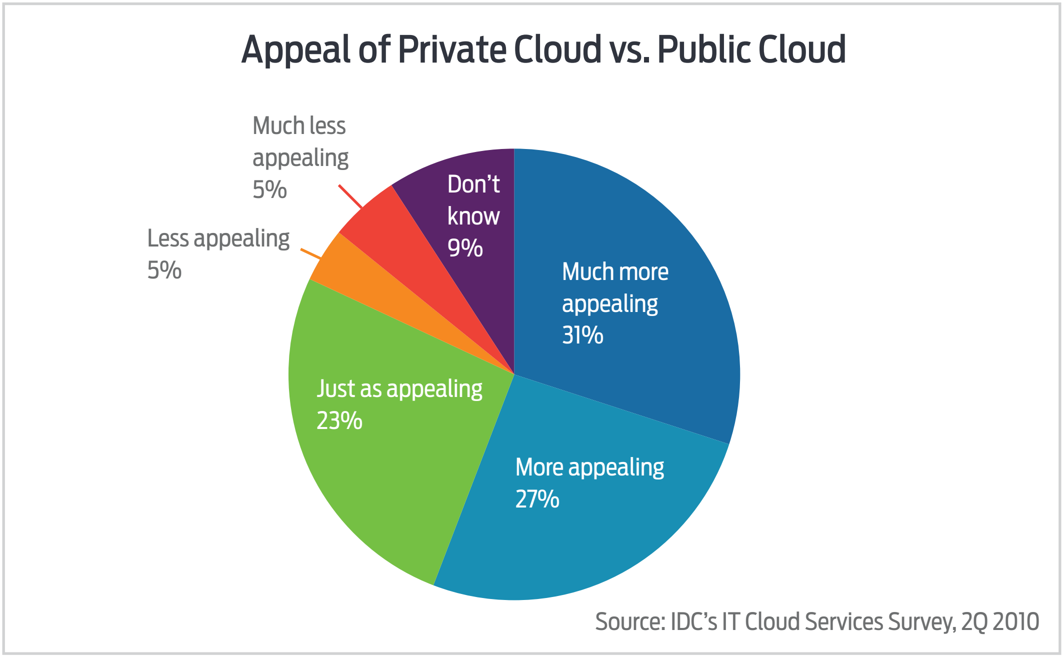
Understanding Private Platform as a Service (PaaS)

PaaS is best and most simply defined as the application container at the center of a revolutionary approach to computing platforms where a full stack–datacenter, OS, runtime, and deployment mechanics–are offered as a service. This means that application developers can write apps that conform to the PaaS's standards, and deploy them to the PaaS with the click of a button. The PaaS offloads a majority of the heavy lifting related to deployment, configuration, and even scaling of a web or SOA application. It also exposes complex workflows associated with application deployment and management to developers through web portals and/or API calls (described in detail later in this whitepaper). PaaS accomplishes this by abstracting away any number of server, OS, and networking components like load balancers into a single resource pool that can be co-habited by multiple "guest applications." In a PaaS, the application components are the first class citizens, while server infrastructure and OS images are merely commodity resources. This is in stark contrast to the current application delivery model where infrastructure and virtual machines (VMs) are the first class citizens in a network, with applications as secondary and subservient to the constraints of the infrastructure.

More advanced PaaS offerings may go beyond simply deploying and managing guest applications, and may also take a critical role in the application execution by routing traffic and even managing memory. Additionally, a PaaS may wrap guest applications with developer operations workflows such as lifecycle management, scale-out capability, and identity management, and may offer cross-cutting services such as caching via an API. Conceptually, this is not much different than traditional application servers - think of PaaS as the application server for the cloud era.

PaaS is typically deployed in a 'public' fashion, meaning that a PaaS provider offers both the PaaS and operates the underlying infrastructure; the entire service is accessible in an "outsourced" fashion to other organizations. While this form factor is good for small development shops, some independent software vendors, and for a number of enterprise use cases, it by no means addresses all, or even a majority of, the needs of enterprise IT. In enterprise IT, there are a myriad of use cases that call for keeping applications behind the firewall, where leveraging a public PaaS is not an option. Constraints such as SLA requirements, latency, regulatory concerns, risk management, and co-located integrations and data access are real concerns that might prevent an enterprise from leveraging public PaaS.

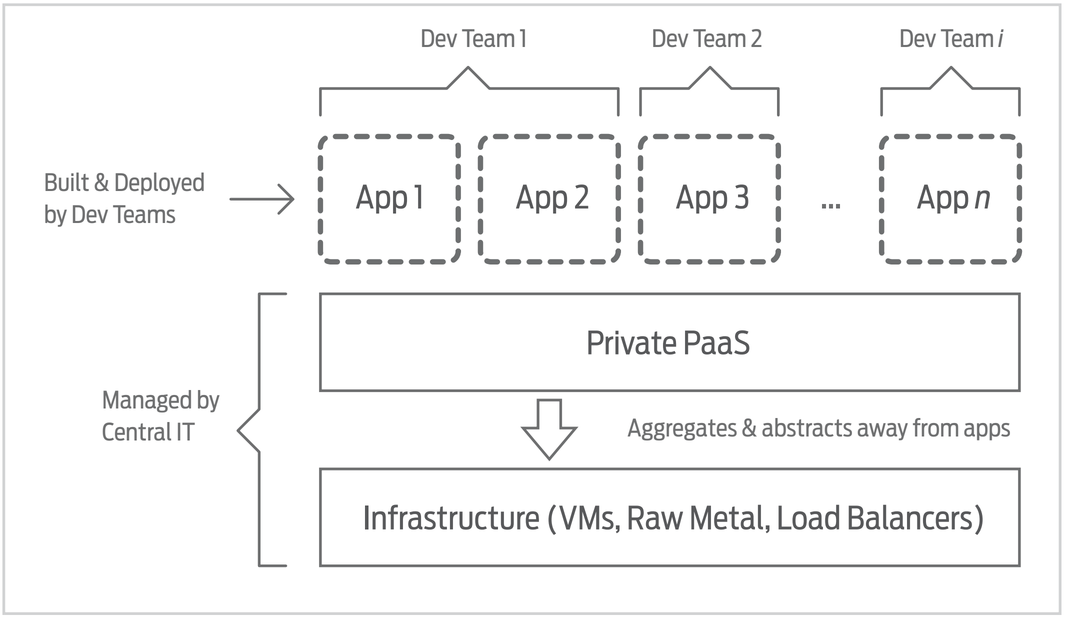
These constraints are pervasive and influence cloud adoption significantly. The CDW 2011 Cloud Tracking Poll of 1,200 IT professionals found that just 7% responded that they would use public cloud services, while 47% responded that they would prefer a private cloud deployment. Similarly, IDC's Q2 2010 IT Cloud Services Survey found that a full 58.4% of respondents found private cloud more appealing than public cloud and 81.7% found private cloud just as appealing or more appealing than public cloud.



Again, this data is not intended to discredit public cloud, but rather to provide credence to the notion that there are real constraints preventing public PaaS adoption. Conceptually, private PaaS is similar to public PaaS from the point of view of development teams and end users. In organizations that have implemented private PaaS, the use case is straightforward.

1. A central/horizontal IT team, the "platform owner," builds or licenses a PaaS layer that it deploys across a cluster of dozens to up to thousands of servers.
2. The platform owner offers the PaaS to internal development teams building horizontal or Line of Business (LOB) applications.
3. Development teams are provisioned to the PaaS, each with their own individual accounts.
4. New applications are written to target the internal PaaS. When ready, the development team either "pushes" the application to the PaaS or uploads the application through a web portal.
5. After some basic configuration, the development team "publishes" the application to the PaaS, which dynamically deploys the application's various components (web services, UIs, databases) to the underlying server cluster managed by the PaaS. Servers are dynamically selected based on the PaaS deployment heuristics.
6. The PaaS monitors and manages the application at runtime.
7. The development team manages application lifecycle workflows through the PaaS.
8. The platform owner never explicitly interacts with the management of an individual application, only with the PaaS itself to ensure its availability, capacity, and general well-being.

Having a PaaS model, whether public or private, creates significant efficiency through abstraction and standardization. Clearly, in the private case, the burden of operating the PaaS is on central IT, but the efficiencies the private PaaS can create far outweigh the nominal cost of operating the PaaS itself. It is important to note that this is a benefit that is truer for private PaaS itself than it is for private cloud in general. Private virtualization clouds (or private Infrastructure as a Service) provide value, but the value may be marginal given that private IaaS squarely focuses on infrastructure flexibility and does little to standardize application development or drive high efficiency enterprise software architectures. Private PaaS's upstack value has the opportunity to manifest much more hard dollar and soft dollar savings than private IaaS given that PaaS touches both infrastructure and development teams.


Dev Team 1
Dev Team 2
Dev Team *i*
Built & Deployed by Dev Teams
App 1
App 2
App 3
...
App *n*
Private PaaS
Managed by Central IT
Aggregates & abstracts away from apps
Infrastructure (VMs, Raw Metal, Load Balancers)

First, all pieces of infrastructure deployed in support of a PaaS are stitched together by the PaaS as a heterogeneous resource pool that manifests itself to consumers as a homogenous platform. This ensures consistency in terms of infrastructure management since the PaaS insulates the platform owner from caring about the details of commodity hardware. Second, applications deployed to the PaaS all conform at some level to the requirements of the PaaS layer, ensuring a level of architecture consistency across development teams and applications. Rather than each application architecture being an exception to the rule, every application follows the rules and, as a result, consistency is achieved. These two pieces of "enterprise DNA," infrastructure abstraction and application standardization, create significant value that will be explored in detail in the next section.

Next Generation Efficiency through Private PaaS

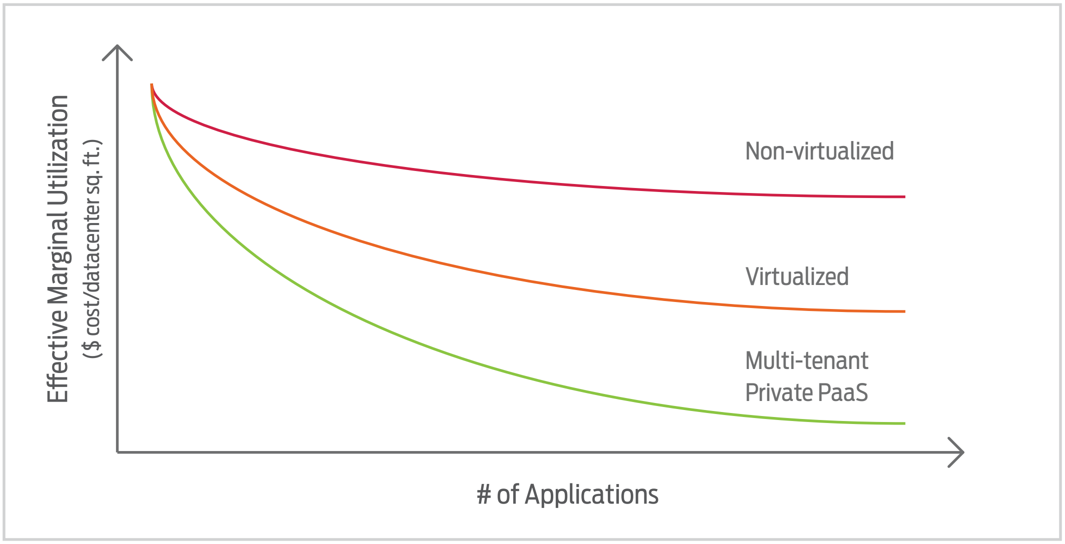
The benefits of private PaaS are many, but thematically they focus on remedying two major problems in enterprise IT:

1. Reducing costs by increasing infrastructure utilization and reducing cycles necessary to manage and maintain application infrastructures.
2. Increasing agility by automating critical "dev-ops" workflows, reducing complexity, and delegating control through self-service.

Having a pragmatic understanding of the key benefits will solidify the value of private PaaS in the context of PaaS's value as a software layer rather than as an outsourcing vehicle.

Increased Infrastructure Utilization

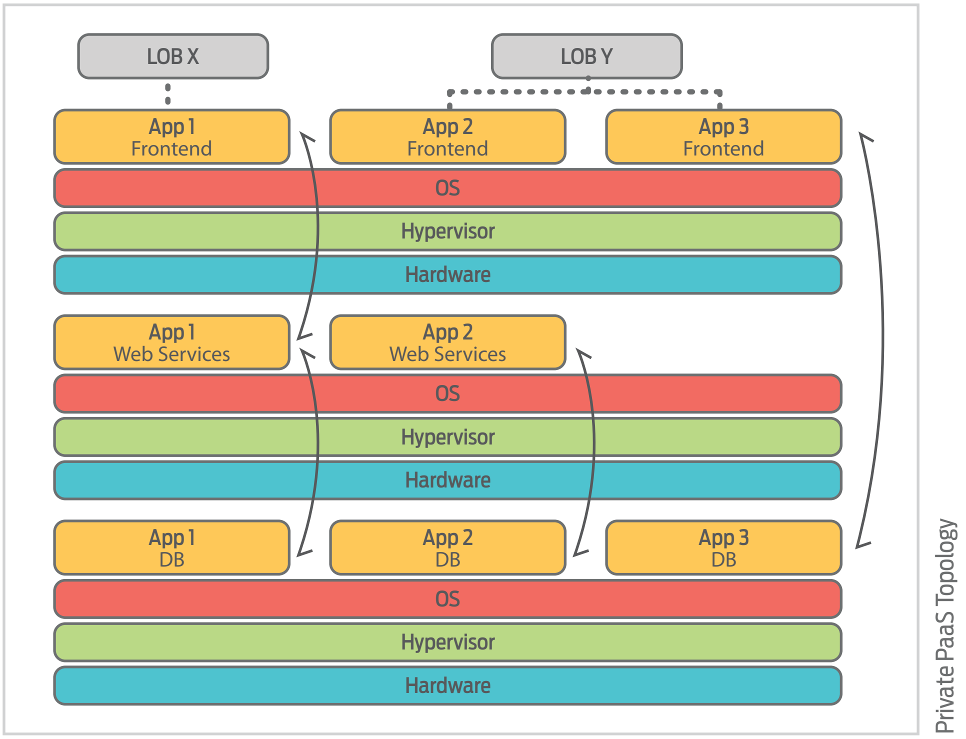
The first point of focus will be to understand private PaaS's impact on infrastructure utilization. Infrastructure utilization is horizontally bound by "isolation units." An isolation unit is best described as the finest grained resource capable of segregating one application from another. Enterprise IT typically uses the OS instance as its finest grained unit of application isolation. Essentially, this means that when an application must be deployed, the best way to ensure that the application is isolated from other applications on the infrastructure is to dedicate an entire OS image, either via a VM or raw metal server, to host that single application. If the application has physically separated tiers, more than one OS instance must be dedicated to the application since each tier may need to be hosted individually.

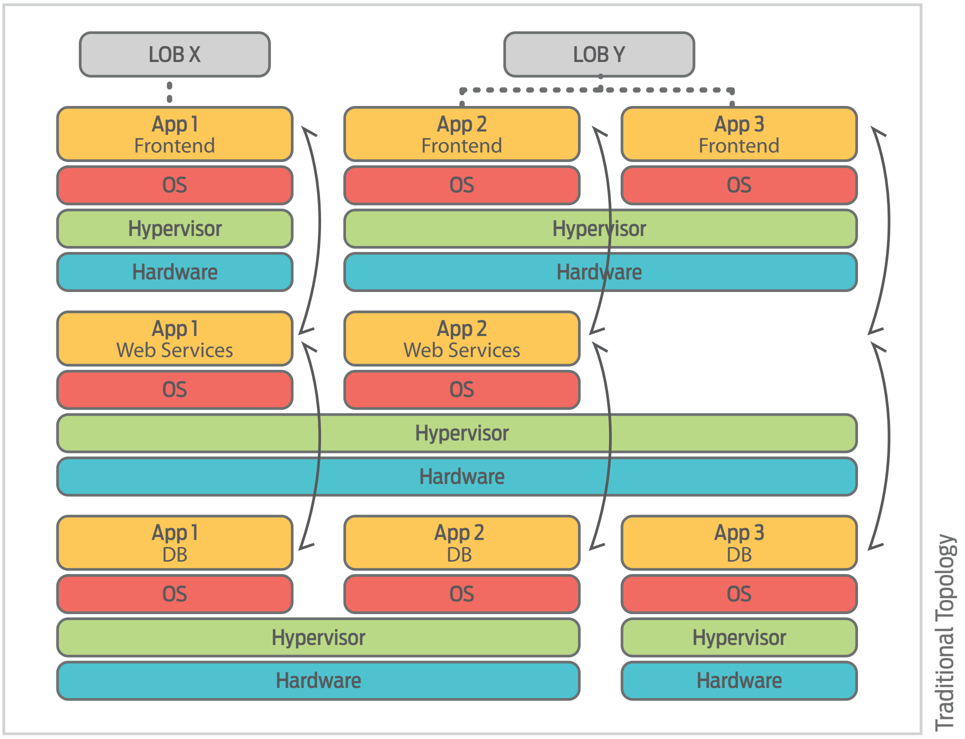


Virtualization somewhat helps with utilization since OS instances can be “overlapped” on shared hardware, boosting utilization. Unfortunately, the utilization boost, at least from a cost point of view, is overstated. Each OS instance may themselves be severely underutilized by the guest application, and in environments and technology stacks where the operating system and databases are licensed, virtualization offers no additional savings as it relates to license costs since a new license must be acquired for each isolation unit.

PaaS technology, if well implemented, is multi-tenant at the operating system level. This means that a PaaS is capable of co-habiting multiple application components on a single OS instance, using processes and memory to “sandbox” the application component within an OS instance alongside other application components. In virtual machine runtimes such as the JVM or Microsoft’s CLR, the JVM or the CLR itself will be used as the isolation unit. This finer grained isolation allows for density levels where up to a few dozen application components can share a single OS instance, rather than each application component having their own OS instance.

To better illustrate this, let us compare and contrast a scenario where three different applications are deployed to an enterprise IT infrastructure, both with and without a multi-tenant PaaS. Two of the applications are 3-tier, with a web frontend, web services, and a database, and the third is 2-tier with just a database and front end. As described earlier, a common approach without a PaaS is to isolate components via individual OS instances resulting in a topology that may look something like this:


LOB X
LOB Y
App 1 Frontend
App 2 Frontend
App 3 Frontend
OS
Hypervisor
Hardware
App 1 Web Services
App 2 Web Services
OS
Hypervisor
Hardware
App 1 DB
App 2 DB
App 3 DB
OS
Hypervisor
Hardware
Private PaaS Topology


LOB X
LOB Y
App 1 Frontend
App 2 Frontend
App 3 Frontend
OS
OS
OS
Hypervisor
Hypervisor
Hardware
Hardware
App 1 Web Services
App 2 Web Services
OS
OS
Hypervisor
Hardware
App 1 DB
App 2 DB
App 3 DB
OS
OS
OS
Hypervisor
Hypervisor
Hardware
Hardware
Traditional Topology

The PaaS takes on the responsibility of intra-OS co-habitation, resulting in an OS licensing count of three licenses rather than eight, or a 62.5% reduction in licensing costs (not including any savings from other infrastructure software licensing such as RDBMS licenses). Furthermore, advanced PaaS offerings will typically add a layer of indirection between tiers to manage application level messaging and route traffic for the application. This creates a highly flexible topology, helping to provide scale-out and high availability.

This elementary analysis quickly shows significant savings with a multi-tenant PaaS approach. An economic dissection of utilization will show even greater efficiencies in licensing utilization, infrastructure utilization, and resource commoditization, which can help achieve up to a ten-fold increase in utilization when compared to non-PaaS environments.

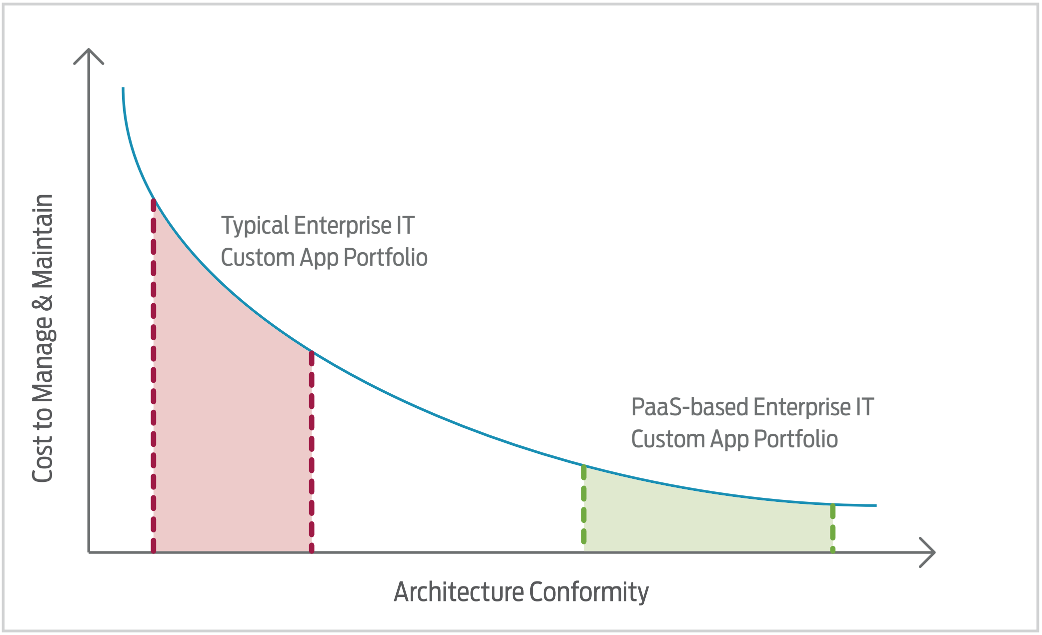
Decreased Application Time to Market & Trivialized Management

Huge advancements have been made in enterprise application development, allowing enterprise developers to write custom SOA and web applications faster than ever. These advancements have allowed Line of Business customers and external business partners to access critical software functionality sooner rather than later. Unfortunately, once a development team has written an application, the process of provisioning an application and having it ready for use tends to be slow and cumbersome.

Typically, a development team can expect anywhere from 10-60 business days, with an average of around 30 days, to have their application provisioned on the infrastructure. During this time period, IT staff will provision a server or VM, prep an OS instance for each application layer, ensure that network dependencies such as DNS entries and load balancer configurations are updated, and, finally, that the application bits are installed and configured properly. If an application took 4 months to develop, we can expect total time to market to be 5-6 months - an increase of 25%-50% over the actual programming phase! Minor changes to the application, such as new releases or necessary infrastructure updates, can each take just as long, adding significant hard-dollar cost, as well as opportunity cost, to an application over its lifetime.

The primary cause of these unfortunate economic profiles is the lack of architecture conformity between the different applications created by the different development teams. This leads to a scenario where each application is an exception to the rule and has different expectations and configuration needs.

In a typical enterprise IT environment, the various development teams likely made different decisions on how to handle concerns such as authentication, authorization, session management, caching, etc. Some may have built subsystems to handle these various cross-cutting application concerns, others may have bought off-the-shelf components, and still others might have used open source components. Because of these permutations, IT cannot automate management or provide broad-stroke policy enforcement. Additionally, developers have no way to expect generalizable value-added services from IT since IT has no architecture expectations of the apps. On change requests, the entire approval process must be repeated and the static nature of the deployed state of the application means that drastic changes may require significant reconfiguration of the application. ovements in the enterprise IT experience without the adoption hurdles associated with public PaaS.



Across the application portfolio, the lack of architecture conformity creates significant costs in managing and maintaining application deployments. Clearly, one solution to this problem is to establish an internal architecture and development standard, but enforcement of this standard is difficult and unlikely, particularly at scale. Furthermore, such a standard typically offers little adoption incentive, and is viewed as a barrier rather than a benefit by developers.

Private PaaS offers a win-win solution that establishes architecture conformity across all guest applications. The PaaS, and not human actors, will provide fundamental workflows such as application provisioning, patch management, and even complex architecture qualities such as scale-out and high availability. IT can manage just the PaaS as an abstraction, and not each individual application. Developers can leverage self-service by deploying to the PaaS, ensuring that they can deploy their applications in 5 minutes rather than 30 days. They can even tap into platform level services such as caching, authentication, and scale-out, drastically simplifying their project workloads for even the most complex task. Having access to powerful platform level functionality acts as a huge incentive for developer adoption. The net result for an enterprise IT organization is that as more and more applications are deployed to a private PaaS, the marginal increase in cost to manage and maintain the PaaS based application portfolio drops drastically.

Commoditize Complex Architecture Patterns & Services

Enterprise application development is a non-trivial task, and typically requires the use of complex architecture patterns and application services to fulfill project requirements. The current development approach leaves each development team to fend for itself. Should a project require in-memory caching, a cache is built or downloaded. This pattern is true for most cross-cutting concerns: logging, authentication, authorization. But what about more complex requirements such as a service bus, or application usage metering, or scalable distributed call patterns like web service broadcasting? These too, are typically “the developer’s problem” to figure out. The net result is that development teams spend significant amounts of time trying to build or source this functionality, causing not only a severe negative impact on time to market, but often a severe impact on software quality if the development team has little to no expertise in creating or integrating those subsystems.

An even deeper problem arises when a development team needs to tackle more complex architectural issues such as high availability and resiliency, scalability, or even modern cloud architecture patterns like multi-tenancy. These architectural facets tend to increase complexity exponentially, slowing down project roadmaps and adding to a project's overall brittleness.

In a private PaaS model, a majority of these headaches go away. PaaS offerings, aside from providing the operating fabric and runtime for a modern cloud application, also aggregate and expose "platform services." Guest applications can either explicitly consume these services via API calls, or implicitly take advantage of them by virtue of being deployed in the PaaS container. Rather than tackling complex development issues, guest applications on a private PaaS can simply expect the PaaS to provide commoditized, shareable standard services. Development teams benefit by being able to narrow their development focus to the business requirements of their project, speeding up development time and improving software quality by accessing well proven, well integrated components. As a result, reductions in development cycle investment of 30%-90% can be typically experienced through leveraging a PaaS.

A private PaaS approach to commoditizing standard architectures and services is much more effective than using shared libraries. Shared libraries tend to require active participation by the development team to adopt since they are not part of a runtime. Much like a Java or .NET application "inherits" memory management by virtue of running in its runtime, PaaS guest applications implicitly "inherit" architectures and capabilities by virtue of running on a PaaS.

Implicit Functionality (Inherited)

- Scale-out architecture (some or all tiers)
- Multi-tenancy (e.g. application level, such as row lovel ownership in a database)
- High availability
- Gateway authentication
- SLA guarantees

ExplicitFunctionality (API Calls)

- Caching
- Job scheduling
- Application usage metering
- Logging
- Authorization
- Service bus/call pattern access

Clearly, not all PaaS functionality is inherited; some requires that an application developer call an API, but the standardization of the API helps drive adoption and reduces maintenance costs.

Much like the application server did for web applications and the OS did for desktop and server applications before it, PaaS, as a runtime and abstraction, helps aggregate common functionality rather than leaving it to development teams to re-invent the wheel each time.

Private PaaS through Apprenda

As with any technology choice, there are two options in private PaaS: build or buy. Apprenda is a deploy anywhere private PaaS that represents the most robust and viable option for developing a PaaS-based private cloud for Microsoft .NET based web and SOA applications.

Created by a team of cloud architecture experts, Apprenda is a private Platform–as-a-Service (PaaS) and framework that allows an enterprise IT organization to deploy their own true multi-tenant PaaS on any infrastructure, making the value of PaaS accessible to all IT use cases. It "brings PaaS to the enterprise" rather than "the enterprise to the PaaS." Through Apprenda, organizations can leverage their expertise and existing investment in Microsoft's .NET, SQL Server, ASP.NET, and Silverlight, and extract value through development and deployment simplicity, automation of cumbersome and costly workflows, and the use of shared architecture services. Apprenda's goal is to revolutionize enterprise IT software delivery in order to save enterprise IT organizations tremendous time and money both upfront and ongoing.

Deployable on-premises or in the cloud, Apprenda stitches together any number of Windows Server, SQL Server, load-balancer, and IIS instances into a single, multi-tenant resource pool for your .NET web and SOA applications. Your applications live as guests on the Apprenda platform and inherit architecture qualities like multi-tenancy, scale-out, and high availability out of the box. Additionally, your applications are seamlessly integrated into Apprenda services that provide mission critical capabilities like application patching, end user provisioning, identity federation, and usage metering - all manageable through point and click web portals.

Apprenda, as a "plug and play" PaaS for your existing infrastructure and custom Microsoft .NET applications, helps realize the value outlined in this whitepaper without the effort and complex engineering required to build a PaaS from scratch.

Conclusion

It is quite clear that cloud, and PaaS in particular, provides significant value. What is also clear, however, is that the public form factor may not be appropriate for a majority of enterprise IT use cases on both the new application and existing custom application front. Even without including the outsourced IT component, the ROI of PaaS is supported by drastic improvements in:

1. Operating agility as it pertains to application deployment and management
2. Infrastructure management
3. Application Development productivity
4. Infrastructure utilization

Coupling these key value benefits to the inaccessible public form factor is unnecessary, and decoupling it has the potential to drive a new era of efficiency within enterprise IT.

We invite you to explore Apprenda at http://www.apprenda.com and learn how your enterprise IT organizations can achieve a next generation IT infrastructure for both IT teams and development teams.