Selecting a Cloud Platform:
A Platform as a Service Scorecard
Chris Haddad  
*VP Technology Evangelism WSO2*

Chris has led successful solution teams building infrastructure and applications supporting Cloud Architecture and Service Oriented Architecture initiatives. Back in the turn of the century (1998-2001), Chris’ team built Platform as a Service and SOA infrastructure supporting the first human resources and benefits management application delivered as a Cloud Native service. The Software as a Service (SaaS) application was delivered to thousands of clients, and lives on as Automatic Data Processing’s mid-size company SaaS offering.

Haddad’s role included building software development teams, creating application platform infrastructure frameworks, establishing repeatable development methodologies, contributing to open source frameworks, hiring and mentoring development talent, liaising with partners, and supporting the production-fielded Software as a Service (SaaS) application.

Prior to joining WSO2, Chris led research and consulting teams at Burton Group and Gartner advising Fortune 500 enterprise organizations and technology infrastructure vendors on adoption strategies, architecture, product selection, governance, and organizational alignment. The team advanced best practices in Platform as a Service, Cloud Application Architecture Patterns, Service Oriented Architecture, and application middleware. While at Burton Group and Gartner, Chris became familiar with PaaS/SOA vendor product roadmaps and strategies. Additionally, his experience with end-user goals, adoption challenges, and constraints guides his architectural recommendations.

Other accomplishments include gaining Apache Axis committer status and speaking at industry events including Gartner Application Architecture Development and Integration (AADI), Gartner Symposium, Burton Group’s Catalyst Conference, Enterprise Architecture Summit, Networld+Interop, Sys-Con SOAWorld, RedHat JBossWorld, OMG Workshops, IASA IT Architecture Regional Conferences, and SDWest.

As Vice President of Technology Evangelism at WSO2, Chris Haddad raises visibility, awareness, and knowledge of the Carbon and Stratos platforms. Chris works closely with developers, architects, or C-level executives within client organizations to increase WSO2 technology adoption, improve the middleware platform, and maximize customer value.
# Table of Contents

Selecting a Cloud Platform: A Platform as a Service Scorecard ......................................................... 1

What is Platform as a Service? ........................................................................................................ 5

**Platform as a Service Attributes** ................................................................................................. 8
  - Measured service or pay per use ................................................................................................. 8
  - Rapid Elasticity ........................................................................................................................... 8
  - Resource Pooling ......................................................................................................................... 9
  - On-demand self-service ............................................................................................................. 9

**Platform as a Service Capabilities** ............................................................................................... 10
  - DevOps Tooling ......................................................................................................................... 10
  - Automated Governance .............................................................................................................. 11
  - Service Level Management ......................................................................................................... 12
  - Consumption Based Pricing ........................................................................................................ 12

**Platform as a Service Topology** ................................................................................................. 13

**Evaluation Framework** ........................................................................................................... 14
  - Cloud Characteristics ................................................................................................................. 15
  - Cloud Dimensions ...................................................................................................................... 15
  - Production Ready ....................................................................................................................... 16
  - DevOps activities and Software Development Life-Cycle phases ........................................... 16
  - Cloud Architecture .................................................................................................................... 17
  - Platform Services ...................................................................................................................... 18
  - Programming Model .................................................................................................................. 19

**Competitive Scorecard** ........................................................................................................... 20

  - Platform as a Service Comparison Chart .................................................................................. 20
  - Platform as a Service Comparison Table .................................................................................... 20

**Quick Start Roadmap** ................................................................................................................ 25

**Key Metrics** ............................................................................................................................... 26

**Quick Start Use Cases** ............................................................................................................. 26
  - DevOps Tooling and On-demand self-service .......................................................................... 26
  - Automated Governance ............................................................................................................. 26
  - Service level management and elastic scale ............................................................................... 27
  - Consumption based pricing and billing ...................................................................................... 27

**Adopting PaaS** ............................................................................................................................ 28
Your organization may be deciding how to migrate applications into the Cloud, and attempting to determine how Cloud enhances your development, operations, and run-time experience. You may have heard how you could:

- Rapidly deliver new capabilities and reduce time to market by
  - Creating application environments in minutes
  - Deploying applications into production in seconds
  - Adopting DevOps practices (e.g. continuous integration, continuous delivery, incremental testing)
  - Re-using existing public and private Cloud services (e.g. data verification, location mapping, news feeds, logistics, authentication)

- Meet user demand by
  - Building distributed, scalable applications
  - Automatically instantiating new service instances
  - Cost effectively scaling applications

- Extend your SOA into the Cloud by
  - Using cloud-native enterprise middleware to easily integrate systems
  - Connecting internal services with external public services
  - Securely bridging applications while meeting service level agreements

Cloud benefits are compelling, and your peers are starting to demonstrate successful test projects, but you realize slick product demonstrations often do not mirror real world complexity. You find your personal Cloud experience challenged by disconnected platform silos, complicated application architecture, diverse infrastructure technologies, and cloud washed services. How can Cloud and Platform as a Service improve a development team’s ability to rapidly deliver high value business applications and meet user demand?
What is Platform as a Service?

Platform as a Service resides within the space between Software as a Service (SaaS) and Infrastructure as a Service (IaaS). IaaS delivers basic network, storage, and compute-processing capabilities as standardized, scalable service offerings. Example IaaS offerings include Amazon EC2/S3, Windows Azure VM Role, and RackSpace Cloud Servers. Software as a Service delivers business software capabilities (e.g. expense reporting, logistics, benefits enrollment) and information feeds as online web applications and web services. Pioneered in the early 2000’s, SaaS was used to by independent software vendors to efficiently deliver an application without requiring on-premise installation, remote updates, and cost prohibitive instance management. Platform as a Service is application middleware offered as a service to developers, integrators, and architects.

Infrastructure as a Service offers development team a bare-bones infrastructure environment, which requires adding middleware, application frameworks, and infrastructure services (i.e. identity, entitlement, application logging). IaaS encapsulates hardware complexity and applies operational best practices. Operation teams create IaaS Clouds by applying virtualization, automation, and standardization to hardware provisioning and allocation tasks. As teams look to apply provisioning and automation to the application platform, interest in DevOps has grown.

The DevOps movement creates a collaborative environment bridging development and operation team members. DevOps enables team members to jointly design, build, and deploy business application and service solutions. The environment closes the gap between business requirements, policies, available runtime resources, and solution development. Development and operation teams use Platform as a Service to design, build, and deliver customized applications or information services. Instead of relying on standardized SaaS, teams using PaaS have more control over solution architecture, quality of service, user experience, data models, identity, integration, and business logic. PaaS offerings often support DevOp practices, which include self-service, automated provisioning, continuous integration, and continuous delivery.

Figure 1 illustrates the Platform as a Service space, which incorporates IaaS DevOps practices and increases solution customization options. IaaS could be considered an unfinished house requiring appliances, cabinetry, and fixtures. At the other spectrum extreme, SaaS offers a fully furnished dwelling with little customization. Even if purple dotted lime décor is not your personal style, a SaaS may require you to sit on the purple dotted lime green couch. Alternatively, a PaaS offers a finished house with an array of personalized furniture choices.
Because the industry broadly defines PaaS as the level above hardware infrastructure and below business applications, development teams do not commonly have clear comparison criteria to intelligently evaluate frameworks or determine adoption benefits. At a minimum, PaaS offerings differ from traditional application platforms by shielding teams from direct infrastructure ownership, management, and complexity. To maximize business benefit, PaaS offerings should significantly exhibit essential Cloud characteristics. The NIST Draft - Cloud Computing Synopsis and Recommendations defines Cloud characteristics as:

- On-demand self-service
- Broad network access
- Resource pooling
- Rapid elasticity
- Measured service

PaaS offerings will vary from cloud washed to cloud native. While cloud washed PaaS solutions will deliver familiar architecture and programming models, they will offer modest Cloud characteristic advancements and deliver only incremental improvement in your ability to meet user demand, rapidly deliver new capabilities, reduce time to market, and lower cost. Cloud native PaaS solutions will inject behavior into the application to decouple application code from run-time infrastructure details, increase application density, and facilitate distributed interactions. To determine whether a PaaS is cloud washed or cloud native, fully evaluate platform capabilities, attributes, and topology.

Cloud washed offerings are optimized for traditional, static single-tenant deployment. Single tenancy will force teams to provision a virtual instance for every client; consuming large amounts of machine resources (e.g. CPU, memory) and increasing administration effort. Cloud washed offerings exhibit elastic scale by provisioning (or de-provisioning) entire physical machines or virtual images, and manual effort is often required to synchronize application platform session state with hardware infrastructure changes. Cloud washed solutions modify the architecture at the virtual machine and network level. The architecture
simply adds hardware virtualization and application platform provisioning. Cloud washed solutions exhibit reference architecture models that closely correlate with those found in traditional, terrestrial on-premise environments. In a Cloud washed environment, application platform components (i.e. application server, enterprise service bus, business process management engine, identity store) are not re-factored to natively support cloud characteristics or supporting architectural attributes (i.e. tenancy, dynamic discovery).

Cloud’s roots reside in hardware infrastructure, IT operations, and web application delivery (i.e. SaaS). Many individuals believe PaaS benefits can be accrued without refactoring the application platform, and few vendors have delivered middleware supporting new application architecture and programming models, which facilitate cloud characteristics. As a result, PaaS offerings often recreate the traditional machine environment in the Cloud. The Cloud washed PaaS environments commonly do not shield application developers, integrators, and architects from infrastructure details (i.e. memory configuration, location, number of machine instances). While short-term benefit is derived by ‘quickly pushing bits into the Cloud’, the design and development experience remains the same.

To improve application and information service delivery, teams must transform the development experience. Cloud native solutions shield teams from infrastructure details and inject new behavior into the application. Cloud native solutions are natively multi-tenant, elastic, dynamically wired, and incrementally testable. The entire application platform stack spanning virtual machine, managed code container (e.g. Java Virtual Machine), application platform engines (e.g. business process engine, presentation engine, enterprise service bus, business activity monitor), and persistence should support multi-tenancy. Multi-tenancy enables teams to customize applications and services per consumer by changing run-time configuration settings instead of provisioning new instances. As user demand grows and shrinks, an elastic and distributed platform will right-size application resources based on utilization, quality of service, and cost optimization policies. When resources are added, subtracted, or moved, the automated service management components dynamically re-wire resource connections. To further increase business agility, the PaaS environment must enable rapid release iterations and incremental solution testing.

This document describes PaaS evaluation criteria, capabilities, and attributes from an application architect’s perspective and architect’s desire to maximize Cloud characteristics while decoupling applications from infrastructure topology specifications.
Platform as a Service Attributes

Cloud characteristics describe measurable service behavior, but the characteristics do not describe how to achieve the behavior. Platform as a Service attributes specify architectural patterns and practices used to realize on-demand self-service, rapid elasticity, resource pooling, and measured service. Figure 2 illustrates the relationship between Cloud characteristics and supporting PaaS attributes.

Figure 2: Cloud characteristics and supporting Platform as a Service attributes

**Measured service or pay per use**

The first Cloud characteristic, measured service, enables pay-as-you-go consumption models and user subscription to metered services. Usage is monitored, and the system generates bills based on charging model. To close the perception gap between business end-users and IT teams, the Cloud solution should bill for business value or business metrics instead of billing for IT resources. Business end-users do not easily correlate business value with an invoice for CPU time, network I/O, or data storage bytes. In contrast, business focused IT teams communicate value and charges based on number of users, processed forms, received marketing pieces, or sales transactions. A cloud native PaaS supports monitoring, metering, and billing based on business oriented entities.

**Rapid Elasticity**

A stateful monolithic application server cluster connected to a relational database does not efficiently scale with rapid elastically. Dynamic discoverability and rapid provisioning can instantiate processing and message nodes across a flexible and distributed topology. Applications exposing stateless services (or where state is transparently cached and available to instances) will seamlessly expand and contract to execute on available resources. A cloud native PaaS will interoperate with cloud management components to coordinate spinning up and tearing down instances based on user, message, and business transaction load in addition to raw infrastructure load (i.e. CPU and memory utilization).
Resource Pooling

Development and operation teams are familiar with resource pooling. Platform environments commonly pool memory, code libraries, database connections, and resource bundles for use across multiple requests or application instances. But because hardware isolation has traditionally been required to enforce quality of service and security, hardware resource utilization has traditionally been extremely low (~5-15%). While virtualization is often used to increase application-machine density and raise machine utilization, virtualization efforts often result in only (~50-60%) utilization. With PaaS level multi-tenancy, deterministic performance, and application container level isolation, an organization could possibly shrink its hardware footprint by half. Sophisticated PaaS environments allocate resources and limit usage based on policy and context. The environment may limit usage by throttling messages, time slicing resource execution, or queuing demand.

Integration and SOA run-time infrastructure supports the resource sharing and interoperability required to deliver effective resource pools. As teams start to pool resources beyond a single Cloud environment, integration is required to merge disparate identities, entitlements, policies, and resource models. As teams start to deliver application capabilities as Cloud services, a policy aware SOA run-time infrastructure pools service instances, manages service instance lifecycle, and mediates access.

On-demand self-service

On-demand self-service requires infrastructure automation to flexibly assign workloads and decrease provisioning periods. If teams excessively customize an environment, they will increase time to market, lower resource pooling, and create a complex environment, which is difficult to manage and maintain. Users should predominantly subscribe to standard platform service offerings, and your team should minimize exceptions.
Platform as a Service Capabilities

Some solution architects find Cloud characteristics and supporting PaaS attributes too abstract and infrastructure focused. Architects may be more interested in delivering measurable business value, shielding IT personnel from complex dependencies, and deliver a productive development and operations (i.e. DevOps) environment. The following PaaS capabilities (See Figure 3) are used to achieve these objectives:

- DevOps Tooling
- Automated Governance
- Service Level Management
- Consumption based pricing

Figure 3: Platform as a Service (PaaS) Capabilities and supporting practices

DevOps Tooling

DevOps tooling creates an environment fostering collaboration between development and operations team members. Practice and tooling enable teams to implement self-service configuration, automated provisioning, policy configuration, and process automation practices which bridge the design, build, deploy, and manage phases within the software development life-cycle. By integrating DevOps tooling with on-demand resource instances, teams can reduce time to market and increase agility.

Cloud computing, PaaS, and DevOps tooling is an opportunity to raise infrastructure abstraction. Figure 4 demonstrates the continuum from hardware infrastructure to business entities. DevOps tooling integrated with PaaS should shield developers from hardware infrastructure concerns and expose business entities.
While Cloud washed PaaS does often facilitates hardware infrastructure configuration, enables rapid infrastructure installation, and delivers machine level access to Cloud machines, the Cloud washed PaaS environment does not shield development team members from hardware infrastructure complexity. Development team members must still be experts in machine sizing, Java Virtual Machine (JVM) configuration, and network topologies.

Some PaaS environments do hide hardware infrastructure concerns and instead expose application platform entities. Application developers can work with familiar application platform entities (i.e. .war files, application frameworks, application sessions), easily install applications on the Cloud, and configure application instances. Cloud environment exposing application platform entities will deliver a familiar application development model and provide the team with a high level of control.

With both the hardware infrastructure and application platform abstraction levels, teams can often forklift existing applications into the Cloud with few modifications. The migrated applications will run in the Cloud, but will not be purpose-built for the Cloud. If Cloud benefits are derived by delivering capabilities ‘as a service’, business entities should be exposed, composed, connected, consumed, and orchestrated as services. A business entity perspective is required to decompose applications and flexibly distribute the entities across Cloud nodes. PaaS environments exposing APIs, services, and communication channels deliver application building blocks at an appropriate abstraction level. Multi-tenancy is extended inside the application, and tenancy can be applied to users, workspaces, and transactions.

**Automated Governance**

Governance is a practice, which defines policies, people, and processes. Effective governance mitigates risks, improves performance, and facilitates correct actions. Automated governance enables application and infrastructure services to efficiently scale across numerous consumers and providers while effectively monetizing, maintaining, and securing assets and consumer-provider interactions. By publishing a service catalogue offering tiered levels of service, teams can promote standard offerings that meet customer requirements. By streamlining access and approval, automated governance encourages customers to choose standard offerings and reduce cost. Scaling a Cloud environment while right-sizing available capacity is
non-trivial, and the infrastructure must support demand management and capacity management activities. When organizations move beyond their first Cloud service release, automated lifecycle management becomes a predominant concern. To effectively manage the service lifecycle, the infrastructure must report on service versions, subscribed consumers, and usage trends. In the run-time environment, an infrastructure authority component makes resource allocation decisions, which are enforced by service level management components.

**Service Level Management**

Service level management enforces governance policies. PaaS infrastructure should integrate service level management activities throughout the solution stack (i.e. network, processing, storage, managed code container, application platform engines, and application logic). Resource monitoring, resource management, performance management, and traffic orchestration must monitor, manage, and optimize machine node instances, message routing, application service location, tenant security, and session state. Intelligent service level management on Cloud native PaaS infrastructure has the ability to raise infrastructure utilization while maintaining quality of service.

**Consumption Based Pricing**

Today, cloud consumption based pricing reflects IT asset monetization (e.g. machine instance per hour, network I/O, storage bytes). However, business users don’t really care how many instances are running in the Cloud. Business users care about business entities, business activity performance, and associated cost. Table 1 below illustrates various pricing units. For example, the number of market leads generated by a marketing piece, or cost to process an insurance policy. Decoupling metering and billing from IT assets and shifting the reporting model to focus on business activity and holistic IT cost will positively change the IT investment conversation. Coupling multi-tenant metering and billing with business activity monitoring and reporting will facilitate the shift.

**Table 1: Exposed Pricing Units**

<table>
<thead>
<tr>
<th>Abstraction Level</th>
<th>Exposed Pricing Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Business Entity or Business Service</td>
<td>User, workspace, transaction, business service, API</td>
</tr>
<tr>
<td>Platform Service</td>
<td>Business process engine, business rules engine, enterprise service bus, gadget server, business activity monitoring, database</td>
</tr>
<tr>
<td>Application Platform</td>
<td>Application session, application framework, application instance</td>
</tr>
<tr>
<td>Hardware Infrastructure</td>
<td>Compute processing, network I/O, storage bytes</td>
</tr>
</tbody>
</table>
Platform as a Service Topology

A Platform as a Service offering should promote deploying applications onto a flexible, distributed topology. To maximize Cloud characteristics, a PaaS should facilitate scaling way out (e.g. across cloud zones, data centers) and automatically distribute fine-grained service component resources. Figure 5 presents a logical view of a cloud application executing across a distributed topology. The Integration Services PaaS service component is used to connect application service components and external cloud services by message passing, not function invocation. Integration services commonly include a Enterprise Service Bus (ESB), service governance registry, service gateways, and message brokers.

Figure 5: Cloudy Topology
Evaluation Framework

As an application platform in the Cloud, PaaS should facilitate Cloud characteristics (i.e. elastic scale, on-demand self service, resource pooling, and consumption based pricing). While all middleware vendors purport to deliver a platform product or service powering the Cloud, all PaaS offerings do not deliver appropriate granularity, abstraction, capabilities, simplicity, and solution breadth. When attempting to achieve Cloud characteristics, development teams often specify the following goals

• Ensure an application satisfies consumer demand while maximizing resource utilization
• Scale workload processing and increase performance while minimizing infrastructure spend
• Allocate, provision, monitor, manage, and administer resources and policies across multiple tenants, nodes, and locations
• Rapidly deploy application and service components on a preferred topology that meets deterministic performance requirements (e.g., replication, utilization, latency, bandwidth, and coherency)

To achieve these goals, PaaS offerings can be evaluated and compared across the following criteria categories:

• **Cloud Characteristics**
  Measures characteristics (i.e. on-demand self-service, resource pooling, rapid elasticity, and measured service) used to distinguish Cloud solutions from traditional application solutions.

• **Cloud Dimensions**
  Measures how widely the solution can be shared (i.e. private, public, community), who is responsible for PaaS environment management (i.e. internal, external), and where the PaaS is located (i.e. on-premise, outsourced) options

• **Production Ready**
  Measures PaaS maturity and suitability for enterprise, mission critical level use

• **DevOps Activities and Lifecycle Phases**
  Measures how to design, construct, deploy, and manage applications and services using DevOps practices (i.e. continuous integration, continuous delivery, automated release management, and incremental testing)

• **Cloud Architecture**
  Measures architecture principles, concepts, and patterns enabling applications to dynamically execute parallel workloads across a highly distributed environment

• **Platform Services**
  Measures how completely the PaaS satisfies development of complex applications by providing comprehensive application middleware components and services
• Programming Model

Measures programming languages and frameworks, which facilitates building applications and services exhibiting Cloud characteristics

Cloud Characteristics

Cloud characteristics define how the Cloud solution differs from traditional, terrestrial infrastructure. As defined by NIST, Cloudy solutions exhibit the following characteristics:

• On-demand self-service
• Resource pooling
• Rapid elasticity
• Measured service or pay per use

On-demand self-service breadth and depth are key evaluation sub-criteria. Can solution architects and end-users subscribe to applications, services, data repositories, and communication channels? Alternatively, is the on-demand self-service capability a lower level mechanism that can reserve and allocate hardware infrastructure (e.g. machine instances, storage)

Resource pools can be reserved on a fixed or dynamic basis. While fixed reservations deliver higher assurance of faster application response, dynamic resource reservation optimizes expense. Just in time resource allocation in conjunction with true measured service where charges are applied only when an application is processing a service request will lower expense. Many organizations pay for warm application instances, which are infrequently used. Sophisticated PaaS environments allocate resources and limit usage based on policy and context. The environment may limit usage by throttling messages, time slicing resource execution, or queuing demand.

PaaS offerings score high (10) when they exhibit cloud characteristics at the appropriate abstraction level (i.e. application component, application service, data set, business process), and score low (1) when they only enable self-service, pooling, elasticity, and pay per use for IT infrastructure assets (i.e. machine, network I/O, storage bytes).

For more information on Cloud characteristics, read the NIST Draft - Cloud Computing Synopsis and Recommendations.

Cloud Dimensions

This evaluation criteria measures how widely the solution can be shared (i.e. private, public, community), who is responsible for PaaS environment management (i.e. internal, external), and where the PaaS is located (i.e. on-premise, outsourced) options. Before selecting PaaS infrastructure, understand how sharing, location, and responsibility impact your decision. Public, private, or community attributes specify how widely the cloud service is shared; a sharing dimension. Internal or external denote the consumer’s view of the Cloud’s service interface. The view is associated with a consumer’s responsibility for service development, operations, and management; a responsibility dimension. A third dimension, on-premise or outsourced, describes where the service assets are located; a location dimension. A hybrid cloud strategy delivers, spans, and connects clouds across all dimension attributes.
Your organization and team will often be confronted with delivering, connecting, and spanning solutions across diverse private and public Cloud environments. Your Platform as a Service architecture and platform service offering should provide a holistic hybrid environment consistently applying governance and policies across internal and external Clouds. Hybrid cloud use cases require interoperability, federation, SOA principles, and infrastructure services to bridge Cloud environments into a unified platform.

PaaS offerings score high (10) when they can run within all dimension coordinates, and score low (1) when they only run within a subset of the dimensions.

For more information on Cloud dimensions, view Chris’ blog post.

Production Ready

This category measures PaaS maturity and suitability for enterprise, mission critical level use. The production ready criteria category measures whether:

- The service provider offers production support warranties
- The PaaS offering services a significant number of paying customers
- The PaaS service maturity
- The Service Level Agreement (SLA) quality
- The Quality of Service (QoS) warranted by the service provider

A production ready PaaS offering will be supported by sophisticated service level management capabilities. The PaaS offering should include the following service level management capabilities:

- Resource monitoring
- Resource management
- Resource quota management
- Performance management
- Traffic orchestration (i.e. message throttling, message routing, message correlation)

PaaS offerings score high (10) when they are fully supported, mature, and fulfill comprehensive service level and quality of service requirements. PaaS offerings score low (1) when they are pre-production releases and do not warrant service levels.

DevOps activities and Software Development Life-Cycle phases

This criteria category measures support for DevOps activities across software development life-cycle phases (i.e. design, develop, test, build, deploy, manage). The category describes the tools and processes used manage application construction and maintenance tasks. Relevant sub-categories include:

- PaaS integration with on-premise software development life-cycle tooling
- PaaS integration with on-premise automated service governance tooling and policy repositories
- Supported DevOps activities (e.g. automated provisioning, self-service configuration, process automation, continuous integration, continuous deployment)
- Automated governance including:
  - Service catalogue and service tiers
  - Demand and capacity management
  - Life-cycle management
PaaS offerings score high (10) when they are well integrated with preferred software development life-cycle tooling across all application life-cycle phases. PaaS offerings score low (1) when they deliver a disconnected and siloed design, development, deployment, and management experience.

Cloud Architecture

This category measures architecture principles, concepts, and patterns enabling applications to dynamically execute parallel workloads across a highly distributed environment. The PaaS architecture model should shield application developers, integrators, and architects from infrastructure, natively support cloud characteristics, and exhibit Cloudy architectural attributes (i.e. tenancy, dynamic discovery). Cloud architectural attributes should span the entire solution stack, and not be constrained to components below the application server. The Cloud Architecture criteria category measures conformance with architecture principles enabling:

Measured service or pay per use

Evaluate the PaaS solution for fine-grained metering, billing, and reporting of business entities, activities, and interactions. For example, can the PaaS meter the number of workspaces created, users added, or business transactions executed?

Rapid elasticity

Mature PaaS offerings scale discrete application service instances rather than scaling monolithic application instances. An ability to rapidly provision small footprint services based on user demand will increase application density and infrastructure utilization. PaaS offerings should exhibit rapid service provisioning, flexible resource allocation, and distributed topologies. Implicit support for shared nothing architecture and the Thirteen Dwarf patterns (e.g. MapReduce) enables application decomposition, parallel processing, and resource coordination.

On-demand self-service

Mature PaaS offerings do more than push application bits to a Cloud node. Mature PaaS offerings perform flexible, run-time workload assignment and automated startup of standard service offerings. Cloud architecture components include a Cloud controller, Cloud service provisioning, and Cloud load balancer.

Resource Pooling

Mature PaaS offerings supply shared everything containers to hosted applications. Multi-tenancy is incorporated throughout the solution stack (i.e. network, processing, storage, managed code container, application platform engines, frameworks, and application logic). Resource utilization is tracked by business transaction or business entity access, and resource allocation conforms to well-defined policies. Distributed caches are used to share resources across entitled tenants while enforcing security.

PaaS offerings score high (10) when they deliver enhanced capabilities at high abstraction and fine granularity. PaaS offerings score low (1) when they do not extend capabilities beyond levels offered by traditional application platform infrastructure.
Platform Services

This category measures how completely the PaaS satisfies development of complex applications by providing comprehensive application middleware components and services. Teams compose platform services to rapidly design, develop, deliver, deploy, and manage applications. Platform services span the following broad categories:

- Presentation services
- Application and service container services
- Business process and business rule services
- Integration services and message brokers
- Composite application services (i.e. mashups and orchestration)
- Complex event processing services
- Data access and persistence services
- Development governance
- Application life-cycle management
- Automated run-time governance services
- Policy registry and repository services
- Identity management
- Security
- Service level management
- Compute, network, and storage infrastructure services

Figure 6: PaaS Platform Services
A complete, unified platform (see Figure 6 above) decreases integration cost, reduces solution complexity, and enables uniformly applying policy and process. PaaS offerings score high (10) when they deliver a comprehensive platform service list and score low (1) when they offer a small subset of the platform required to effectively deliver complex applications and services.

**Programming Model**

The programming model category measures programming languages and frameworks, which facilitates building applications and services exhibiting Cloud characteristics. The Cloud programming model is fundamentally different from the web programming model which fueled Java EE’s popularity. To maximize cloud benefits, development teams must adopt new programming models, architecture patterns, and frameworks. The actor model, service composition, RESTful interactions, policy injection, and tenant aware frameworks extend Java into the Cloud.

A Cloud programming model facilitates building applications and services, which exhibit Cloud characteristics. The programming model should be congruent with Cloud architecture principles and patterns. Example detailed criteria include:

- Actor model (i.e. message passing instead of function invocation)
- RESTful interactions
- Dynamic recoverability
- Consensus protocols
- Asynchronous rather than synchronous interactions
- Shared nothing architecture
- Data partitioning and sharding
- Federated data queries
- Service orchestration
- Functional programming
- MapReduce

PaaS offerings score high (10) when the programming model explicitly supports parallel processing, distributed interactions, shared nothing architecture, workload decomposition, and hide infrastructure complexity. PaaS offerings score low (1) when they expose data center infrastructure concepts (i.e. machines, storage parameters, network addresses) and do not elastically scale beyond traditional application server clusters.
Competitive Scorecard

A competitive scorecard can help enterprise architects and solution architects evaluate and select PaaS offerings. The scorecard can be used to shortlist PaaS providers, build questions posed in a Request for Proposal (RFP) document, or generate use case scenarios. The scorecard contains comparison tables listing detailed criterion and a comparison chart visually illustrating strengths and weaknesses.

Platform as a Service Comparison Chart

A graphical comparison chart (see Figure 7) demonstrates similar aggregate scores for most Java PaaS providers. While the aggregate scores may be similar, each product has unique strengths and weaknesses. The category bar lengths illustrate a wide variation across criteria categories.

Figure 7: PaaS Comparison Scorecard

Platform as a Service Comparison Table

The Platform as a Service comparison table scores delineate PaaS provider strengths and weaknesses. A score of zero indicates low conformance with the criterion goal state, and a score of ten indicates high conformance. Table 1 provides an aggregate view across the seven evaluation criteria categories. The aggregate scores are a simple average of the detailed criteria. No provider scores more than 50% of the total available points; demonstrating market immaturity. Table 2 provides detailed scores across all 84 evaluation criteria.
### Table 1: Platform as a Service Comparison Table Version 1.0 – Aggregate View

<table>
<thead>
<tr>
<th></th>
<th>WSO2 Stratos</th>
<th>Google App Engine</th>
<th>Amazon Beanstalk</th>
<th>Heroku</th>
<th>CloudBees RUN@</th>
<th>Red Hat OpenShift</th>
<th>VMWare CloudFoundry</th>
<th>Oracle Public Cloud</th>
<th>IBM Application Services</th>
<th>Apprenda SaaSGrid</th>
<th>Windows Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cloud Characteristics</strong></td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td><strong>Cloud Dimensions</strong></td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>Production Ready</strong></td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
</tr>
<tr>
<td><strong>DevOps activities and phases</strong></td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
</tr>
<tr>
<td><strong>Cloud Architecture</strong></td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td><strong>Platform Services</strong></td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td><strong>Programming Model</strong></td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td><strong>Total Points (70 maximum)</strong></td>
<td>29</td>
<td>29</td>
<td>18</td>
<td>18</td>
<td>19</td>
<td>17</td>
<td>18</td>
<td>14</td>
<td>18</td>
<td>30</td>
<td>27</td>
</tr>
</tbody>
</table>

### Table 2: Platform as a Service Comparison Table – Detailed View

<table>
<thead>
<tr>
<th>CIO and architect level view</th>
<th>WSO2 Stratos</th>
<th>Google App Engine</th>
<th>Amazon Beanstalk</th>
<th>Heroku</th>
<th>CloudBees RUN@ Cloud</th>
<th>Red Hat OpenShift</th>
<th>VMWare CloudFoundry</th>
<th>Oracle Public Cloud</th>
<th>IBM Application Services</th>
<th>Apprenda SaaSGrid</th>
<th>Windows Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cloud Characteristics</strong></td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>6</td>
<td>5</td>
</tr>
<tr>
<td>On-demand self-service</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>Resource pooling</td>
<td>7</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>Rapid elasticity</td>
<td>5</td>
<td>8</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Measured service or pay per use</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td><strong>Cloud Dimensions</strong></td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Sharing</td>
<td>7</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>private</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>public</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>community</td>
<td>7</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>Responsibility</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>internal</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>external</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>hybrid</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Location</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>5</td>
<td>3</td>
</tr>
<tr>
<td>on-premise</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>CIO and architect level view</td>
<td>WSO2 Stratos</td>
<td>Google App Engine</td>
<td>Amazon Beanstalk</td>
<td>Heroku RUN@ Cloud</td>
<td>CloudBees</td>
<td>Red Hat OpenShift</td>
<td>VMWare Cloud-Foundry</td>
<td>Oracle Public Cloud</td>
<td>IBM Application Services</td>
<td>Apprenda SaaSGrid</td>
<td>Windows Azure</td>
</tr>
<tr>
<td>---------------------------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>-------------------</td>
<td>-----------</td>
<td>-------------------</td>
<td>----------------------</td>
<td>---------------------</td>
<td>-------------------------</td>
<td>-----------------</td>
<td>---------------</td>
</tr>
<tr>
<td>outsourced</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Production Ready</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Production support warranties</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Number of paying customers</td>
<td>2</td>
<td>8</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Current service maturity</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Service Level Agreement (SLA) quality</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Warranted Quality of Service (QoS)</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Service level management capabilities should include:</td>
<td>6</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Resource monitoring</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Resource management</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Resource quota management</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Performance management</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Traffic orchestration and throttling</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>DevOps activities and Software Development Life Cycle phases</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Integration with on-premise software development life cycle tooling</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>2</td>
<td>6</td>
</tr>
<tr>
<td>Integration with on-premise automated service governance tooling and policy repositories</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Supported DevOps activities (e.g. automated provisioning, self-service configuration, process automation, continuous integration, continuous deployment)</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Automated governance including:</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Service catalogue and service tiers</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>4</td>
</tr>
<tr>
<td>Demand and capacity management</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>CIO and architect level view</td>
<td>WSO2 Stratos</td>
<td>Google App Engine</td>
<td>Amazon Beanstalk</td>
<td>Heroku</td>
<td>CloudBees RUN@ Cloud</td>
<td>Red Hat OpenShift</td>
<td>VMWare Cloud-Foundry</td>
<td>Oracle Public Cloud</td>
<td>IBM Application Services</td>
<td>Apprenda SaaSGrid</td>
<td>Windows Azure</td>
</tr>
<tr>
<td>--------------------------------</td>
<td>--------------</td>
<td>-------------------</td>
<td>------------------</td>
<td>--------</td>
<td>----------------------</td>
<td>-------------------</td>
<td>---------------------</td>
<td>----------------------</td>
<td>-------------------------</td>
<td>-----------------</td>
<td>--------------</td>
</tr>
<tr>
<td>Lifecycle management</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Infrastructure Authority integration</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>lifecycle phases (i.e. design, develop, test, build, deploy, manage)</td>
<td>4</td>
<td>2</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

| Cloud Architecture         | 5            | 5                 | 3                | 3      | 3                    | 3                 | 3                   | 2                    | 2                       | 5              | 4            |

|Measured service or pay per use| 5            | 3                 | 2                | 3      | 3                    | 3                 | 2                   | 2                    | 4                       | 3              |             |
|Fine-grained metering        | 6            | 3                 | 1                | 3      | 3                    | 3                 | 3                   | 2                    | 2                       | 4              | 2            |
|Billing                      | 5            | 2                 | 2                | 2      | 2                    | 2                 | 2                   | 2                    | 2                       | 5              | 4            |
|Reporting                    | 3            | 3                 | 3                | 3      | 3                    | 3                 | 3                   | 3                    | 3                       | 3              | 3            |

|Elastic and Scalable         | 4            | 6                 | 3                | 3      | 3                    | 3                 | 3                   | 2                    | 2                       | 5              | 4            |

|Stateless services           | 4            | 7                 | 1                | 3      | 3                    | 3                 | 3                   | 1                    | 3                       | 5              | 5            |
|Rapid provisioning           | 5            | 7                 | 5                | 5      | 5                    | 5                 | 5                   | 2                    | 3                       | 5              | 5            |
|Deterministic performance    | 2            | 5                 | 2                | 2      | 2                    | 2                 | 2                   | 2                    | 1                       | 2              | 2            |
|Flexible topology            | 7            | 7                 | 5                | 2      | 2                    | 2                 | 2                   | 1                    | 2                       | 7              | 7            |
|Distributed cache            | 2            | 6                 | 2                | 1      | 1                    | 1                 | 1                   | 6                    | 2                       | 1              | 4            |
|Thirteen dwarf patterns (e.g. MapReduce) | 1            | 1                 | 1                | 1      | 1                    | 1                 | 1                   | 1                    | 1                       | 1              | 1            |
|Parallel processing          | 1            | 1                 | 1                | 1      | 1                    | 1                 | 1                   | 1                    | 1                       | 3              | 3            |

|On-demand self-service       | 5            | 6                 | 3                | 4      | 4                    | 4                 | 4                   | 2                    | 3                       | 6              | 6            |
|Flexible workload assignment | 5            | 7                 | 2                | 3      | 3                    | 3                 | 3                   | 2                    | 2                       | 5              | 5            |
|Standard service offerings   | 6            | 4                 | 2                | 4      | 4                    | 4                 | 4                   | 2                    | 2                       | 6              | 6            |
|Quick startup and automation | 5            | 7                 | 5                | 5      | 5                    | 5                 | 5                   | 3                    | 5                       | 7              | 7            |

|Shared, virtual infrastructure and resource pooling | 4          | 4                 | 2                | 3      | 3                    | 3                 | 3                   | 2                    | 2                       | 4              | 4            |

<p>|Multi-tenancy                | 4            | 6                 | 1                | 2      | 2                    | 2                 | 2                   | 2                    | 1                       | 1              | 4            |
|Resource utilization         | 5            | 8                 | 1                | 3      | 2                    | 2                 | 2                   | 2                    | 2                       | 2              | 5            |
|Resource pooling             | 2            | 2                 | 2                | 2      | 2                    | 2                 | 2                   | 2                    | 2                       | 2              | 5            |
|Shared nothing architecture  | 3            | 5                 | 1                | 2      | 2                    | 2                 | 2                   | 2                    | 1                       | 1              | 3            |
|Portability                  | 3            | 1                 | 2                | 3      | 3                    | 3                 | 6                   | 3                    | 3                       | 4              | 4            |
|Interoperability             | 4            | 1                 | 3                | 3      | 5                    | 5                 | 5                   | 3                    | 3                       | 4              | 4            |</p>
<table>
<thead>
<tr>
<th>CIO and architect level view</th>
<th>WSO2 Stratos</th>
<th>Google App Engine</th>
<th>Amazon Beanstalk</th>
<th>Heroku</th>
<th>CloudBees RUN@ Cloud</th>
<th>Red Hat OpenShift</th>
<th>VMWare Cloud-Foundry</th>
<th>Oracle Public Cloud</th>
<th>IBM Application Services</th>
<th>Apprenda SaaSGrid</th>
<th>Windows Azure</th>
</tr>
</thead>
<tbody>
<tr>
<td>Platform Services</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Presentation services</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Application and service container services</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>6</td>
<td>4</td>
<td>6</td>
<td></td>
</tr>
<tr>
<td>Business process and business rule services</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>Integration services and message brokering</td>
<td>6</td>
<td>2</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Composite application services (i.e. mashups, orchestration)</td>
<td>6</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Complex event processing services</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>Data access and persistence services</td>
<td>5</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>6</td>
<td>4</td>
</tr>
<tr>
<td>Development governance</td>
<td>2</td>
<td>1</td>
<td>0</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>2</td>
<td>6</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Application lifecycle management</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Automated runtime governance services</td>
<td>4</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Policy registry and repository services</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Identity management</td>
<td>5</td>
<td>4</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Security</td>
<td>4</td>
<td>4</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Service level management</td>
<td>2</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>3</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Compute, network, and storage infrastructure services</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Programming Model</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>2</td>
</tr>
<tr>
<td>Actor model (i.e. message passing instead of function invocation)</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dynamic recoverability</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
</tr>
<tr>
<td>RESTful interactions</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
</tr>
<tr>
<td>Consensus protocols</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Shared nothing architecture</td>
<td>3</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Asynchronous rather than synchronous interactions</td>
<td>2</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>2</td>
</tr>
</tbody>
</table>
Quick Start Roadmap

When your team is selecting a PaaS provider, road test the PaaS offerings. At a minimum, become comfortable with your team’s ability to realize goals, measure improvements using key metrics, and confidently design, build, deploy, and manage cloud-enabled applications.

Organizations commonly set agility goals (e.g. rapidly deliver new capabilities, reduce time to market), economic goals (e.g. cost effectively scale environment to meet business demand, avoid operating expenses), or efficiency goals (e.g. re-use existing investments, increase operational efficiency, reduce data center footprint). Figure 8 provides a graphical illustration.

Figure 8: Common PaaS goals
Key Metrics

Agreeing on a strategic goal and gaining stakeholder buy-in is only the first step. Teams should define key metrics, baseline current performance, and create a quick start project plan to prove PaaS benefits. Metrics can be divided into foundational, optimal, and transformational categories. A few example metrics to consider are:

- **Foundation**
  - Time to create new application environment
  - Time to redeploy application
- **Optimize**
  - Minimum and maximum scale
  - Scale frequency (i.e. time to scale up/down)
- **Transformation**
  - Time and effort required integrating business process, event processor – creating a complex app.
  - Time and effort required to apply policy across tenant(s)
  - Cost to operate application per user or transaction

Quick Start Use Cases

In today’s IT environment, demonstrating tangible improvement is often illusive. Rather than simply stating a goal, randomly selecting a PaaS provider, installing an application on the PaaS, and declaring success, your team has an opportunity to demonstrate and prove how PaaS capabilities can improve agility, efficiency, or platform economics. Your team should correlate quick start demonstration use cases with PaaS capabilities and key metrics. The following listing details a sample plan:

**DevOps Tooling and On-demand self-service**

- **Use Cases**
  - Rapidly provision application environment
  - Rapidly provision application tenant
  - Allocate, provision, monitor, manage, and administer resources across multiple tenants, nodes, and locations
  - Develop complex, composite integrated applications
- **Key metrics**
  - Time to create new application environment
  - Time to redeploy application

**Automated Governance**

- **Use Cases**
  - Create users and configure rights
  - Deploy on preferred topology that meets deterministic performance requirements (e.g., replication, utilization, latency, bandwidth, and coherency)
  - Create service throttling and security governance (XACML) policies
- **Key Metrics**
  - Time and effort required integrating business process, event processor – creating a complex app.
  - Time and effort required to apply policy across tenant(s)
Service level management and elastic scale

- Use cases
  - Ensure application satisfies consumer demand while maximizing resource utilization
  - Scale workload processing and increase performance while minimizing infrastructure spend
  - Load test application service
  - Demonstrate multi-tenant web application
- Key Metrics
  - Minimum and maximum scale
  - Scale frequency (i.e. time to scale up/down)

Consumption based pricing and billing

- Use cases
  - View service logs
  - View bill by business value
- Key Metric
  - Cost to operate application per user or transaction
Adopting PaaS

Cloud can deliver incremental benefits or provide a transformational opportunity. To maximize innovation and transformation, shield development teams from hardware infrastructure details, raise the level of abstraction, and inject new behavior into the application. Choose Cloud native solutions that are natively multi-tenant, elastic, dynamically wired, and incrementally testable. Use the evaluation framework and quick start roadmap as guides assisting your choice of goals, key metrics, and use cases.

Start your PaaS journey slowly by performing proof of concept quick start projects, which demonstrate clear business value and test use cases. Monitor continued benefit as you scale adoption throughout the organization with a focused marketing and communication plan. When establishing a Platform as a Service roadmap, fully consider cultural and educational hurdles, which may challenge adoption. For example, highly elastic and scalable performance may require teams to re-write applications and adopt new architecture guidance. Behavior doesn’t usually change quickly, and each participant will be asking ‘How does the change benefit me’? Take time to chart challenges, constraints, and marketing plans in your Platform as a Service strategy.

Don’t create a siloed Platform as a Service strategy, but link the strategy with complementary initiatives (i.e. agile, Service Oriented Architecture, Master Data Management, Business Intelligence) and build upon their traction. Adopting PaaS will require an iterative approach; enjoy your journey from the trenches to the stratosphere.

For more information about WSO2 products and services, please visit http://wso2.com or email bizdev@wso2.com