

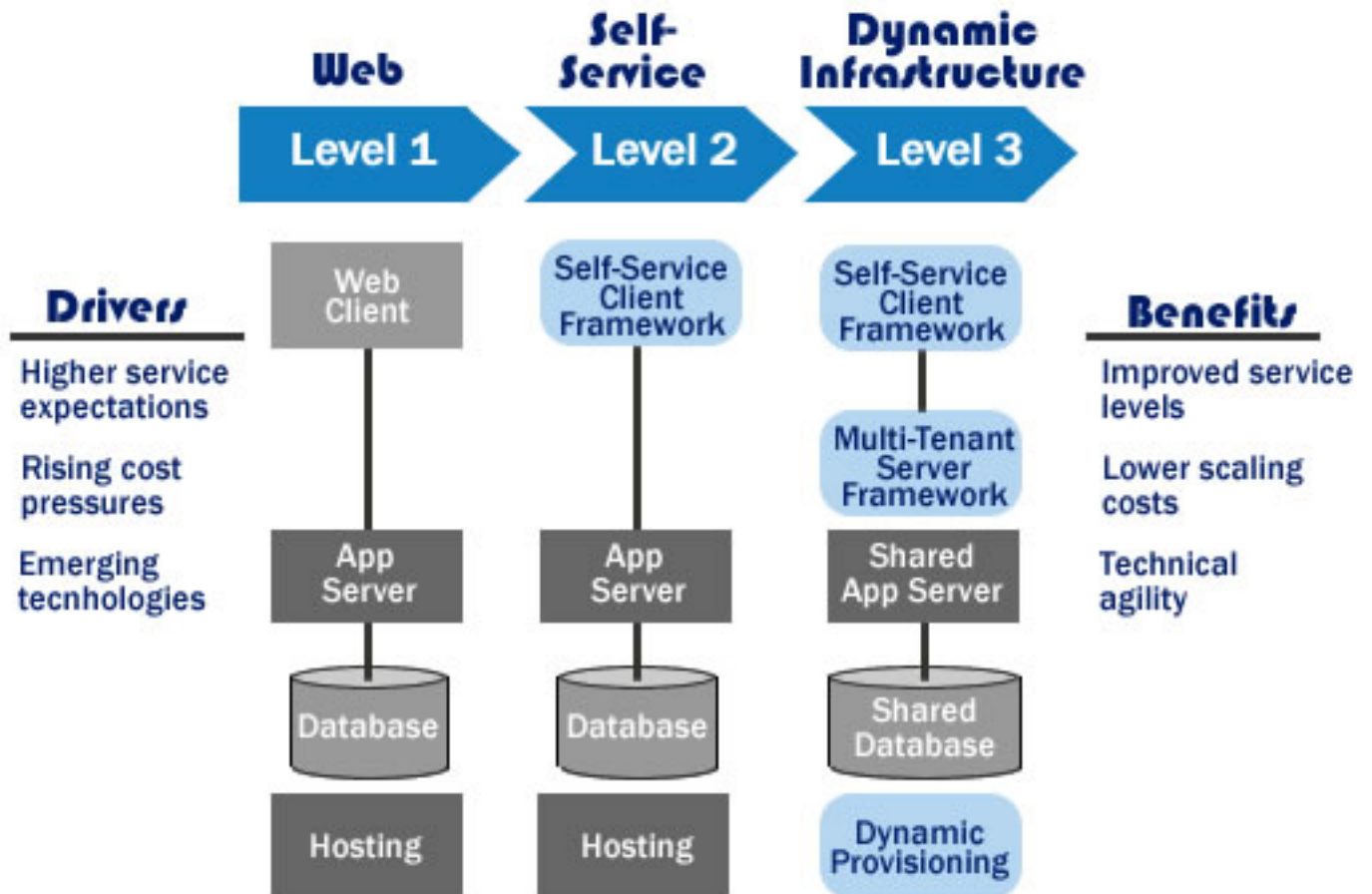
Cloud Quick Start: A Roadmap For Adopting Cloud Computing

by Chris Keene, WaveMaker (ckeene @ wavemaker.com)
Indrajit Poddar, IBM (ipodar @ us.ibm.com)
Joe Nicke, IBM (jnicke @ us.ibm.com)
Uri Budnik, RightScale (uri @ rightscale.com)

This document describes how ISVs can build applications that take advantage of SaaS and Cloud technologies. The focus of this document is on a flexible roadmap for migrating to cloud computing that delivers maximum benefit to the ISV at minimum cost and risk.

The cloud quick-start program is joint enablement engagement delivered by IBM, WaveMaker, and RightScale that builds on best practices developed by IBM's SaaS and Cloud Computing specialty groups. This program helps ISVs assess the readiness of their applications for web-delivery and includes a hands-on cloud proof of concept.

Cloud Quickstart Roadmap



Cloud Quick Start Roadmap

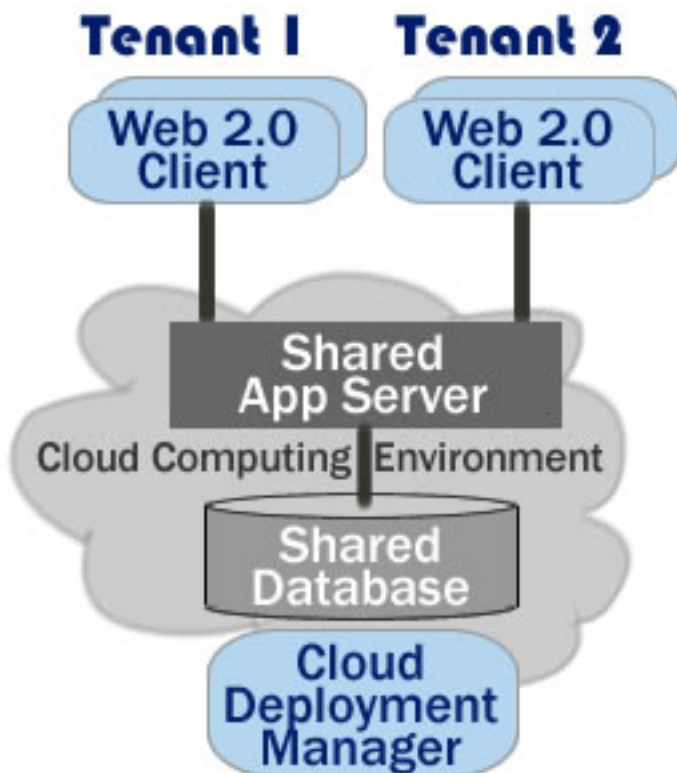
The best practice for ISVs to adopt cloud and SaaS computing is to follow the cloud quick start roadmap. The following figure shows the cloud quick start roadmap:

ISVs should choose the roadmap level that provides the maximum business value for the minimum cost and risk to the ISV. The milestones in the Cloud quick start Roadmap are:

- **Level 1: Web architecture.** Adding a rich user interface, for example by using Ajax technologies like Dojo, provides immediate customer value by enabling distributed application to the application for any end user from their browser.
- **Level 2: Self-service architecture.** Adding client-side frameworks that facilitate the end-user's ability to perform self-service customization of the application gives the end user valuable business flexibility while also reducing their costs for making changes to their business applications.
- **Level 3: Dynamic infrastructure.** Architecting an application to provide shared access to the application and database layer delivers scalability and resilience to the ISV, but may require significant changes to application logic. However, once an application is architected to support shared application and database multi-tenancy, hosting the application on a cloud infrastructure can provide optimal price performance with built-in fault tolerance.

A Multi-Tenant Architecture Example

The following figure shows an example of a SaaS and cloud enabled ISV application. In this approach, multiple customers or tenants share a single instance of the application and also share a single database server instance.



This application example follows "Approach 1: Shared middleware with a single application instance", as described in the IBM developerWorks article titled "Develop and Deploy Multi-Tenant Web-

delivered Solutions using IBM middleware: Part 2: Approaches for enabling multi-tenancy" (<http://www.ibm.com/developerworks/webservices/library/ws-multitenantpart2/index.html>).

Because every ISV has different business and technical requirements, the shared everything architecture is not always the optimal approach. The cloud quick start roadmap is meant to be customized on a per ISV basis as part of the cloud quick start program. For more information on this program, please contact Joe Nicke (jnicke@us.ibm.com).

Business Drivers for Cloud and SaaS Computing

While SaaS and Cloud computing are receiving a great deal of media attention today, there are underlying business drivers forcing all ISVs to develop SaaS offerings. These drivers include increased value to end customers and lower costs to vendors.

For customers, software as a service is synonymous with low upfront costs, easy installation and painless application scaling. Unfortunately, SaaS applications are also commonly held to be inflexible and hard to customize.

For ISVs, cloud computing offers a vision of low cost scalability to support new customers and even lower systems administration costs. However, the cost of migrating an existing ISV application to SaaS may more than offset these savings.

ISV Cloud Readiness Assessment

For an ISV, there are significant customer benefits and cost savings in adopting SaaS and cloud computing. Yet there are substantial costs and risks as well. Ultimately, the optimal path to cloud computing depends heavily on the ISVs existing architecture, code base and data model.

The following levels represent a migration path that can span from from client/server to multi-tenant cloud computing. The first step in a cloud migration plan is to determine where your existing products align against the 3 cloud readiness levels.

- **Level 1: web architecture.** A level 1 application has a rich, interactive web user interface.
- **Level 2: self-service architecture.** A level 2 application has a web user interface that can be configured by end-users without coding.
- **Level 3: dynamic infrastructure.** A level 3 application has been architected to support multiple tenants sharing sharing application and database servers and has the ability to automatically scale to meet increased usage, can transparently failover and recover from server failures.

Technical Challenges for ISVs

Having determined how your application lines up against the cloud readiness levels, the next step for the ISV is to decide which cloud readiness level they would like to reach. This requires weighing the customer and cost benefits of each cloud readiness level against the associated development costs and risks.

Key technical challenges for the ISV include:

- **Multi-tenant Isolation:** how can I keep data and logic separate and secure between tenants? Do I have a robust security, user and role management framework in place? Do I have the ability to extend the data model without changing the schema?
- **End-user Self-Service:** how can tenants make changes to their data model and logic? Can end-users configure their own dashboards, extend the data model, change business workflows on the fly?

- **Application Migration:** how can I use my existing code and data in a multi-tenant architecture? Is there a way to reuse my business logic?
- **Open Platform:** how can I avoid being locked in to a proprietary platform? Can I host my application with multiple service providers or just one? Do I have the option to run my application in an open source environment if I choose?

Managing Cost And Risk In Migrating To Cloud Computing

The best practice approach to managing cost and risk in migrating to cloud computing is to create a roadmap which defines an action plan for moving from your current cloud readiness level to your desired cloud readiness level. With this roadmap in hand, an ISV can optimize their migration to cloud and SaaS computing.

The value of the cloud quick start roadmap is that it prescribes a stepwise approach to migrating ISV applications to cloud and SaaS computing. Rather than rearchitecting the entire application from scratch, ISVs can leverage existing logic and data.

Best Practice Cloud Client Requirements

In migrating to SaaS and cloud computing, much of the success of the application depends on the attractiveness and flexibility of the user interface. In particular, best practice SaaS applications enable end-users to configure elements of the application without coding.

Here are best practices requirements to consider in selecting a cloud client framework:

1. **Select an open, standards-based client technology such as the Dojo Toolkit.** In general, any UI that requires a proprietary client plug-in introduces browser dependencies and locks an ISV into a single vendor. The Dojo Toolkit is a good choice as it is an open source project heavily supported by IBM.
2. **Provide end-user personalization capabilities.** End users have been trained by Web 2.0 applications like MyYahoo and FaceBook to expect that the application user interface will be under their control. A drag and drop framework for enabling end users to configure their own dashboards is essential for a competitive SaaS offering.
3. **Plan for internationalization and accessibility requirements.** Any new Web 2.0 offering should be architected to be able to support internationalization and accessibility requirements.
4. **Enable visual configuration of key business workflows.** For advanced users, the ability to change certain workflows without requiring expert coders can greatly improve their business agility. For example, KANA software considers their browser-based, visual workflow manager as the "killer feature" for their call center solution.
5. **Manage user access to widgets and events based on user roles.** A robust and flexible client security framework is critical for any SaaS application, which has to ensure security not just within a single enterprise but across enterprise tenants.
6. **Manage user access to data from the server side.** A browser client cannot be considered fully secure. Therefore, there must be a strong integration between server side data security framework and the client security framework, such that no client can get access to data for which they are not authorized.
7. **Allow tenants to customize the application template.** Each tenant should have the ability to customize the basic look and feel of their application, for example by adding their own logos and color schemes.
8. **Give tenants the ability to create and import custom widgets.** One way to increase the flexibility of a SaaS application is to make it easy for users to add custom widgets that extend the functionality of their application and minimize the cost and time for integrating multiple SaaS applications.

Best Practice Cloud Server Requirements

Reducing operating costs through cloud computing depends on the ability to share a pool of application and data resources across multiple tenant customers. Typically, this requires specialized server-side frameworks for secure sharing of server-side resources.

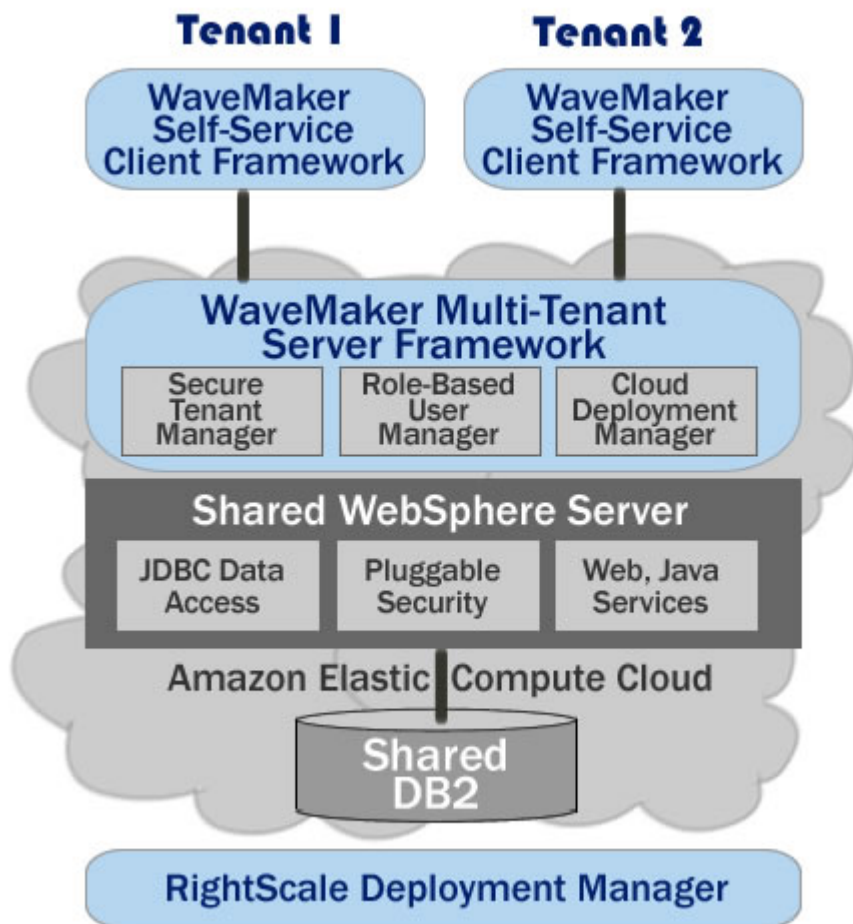
Here are best practices requirements to consider in selecting a multi-tenant server framework:

1. **Select an open, standards-based server framework.** Many Platform as a Service (PaaS) offerings are based on proprietary languages and hosting providers. ISVs should choose an open, Java-based framework for SaaS applications.
2. **Ensure secure tenant access to shared data.** A robust SaaS application requires secure data access to ensure each user sees only data that belongs to their tenant. Best practice design gives each table a tenant identifier column and filters all queries using that identifier.
3. **Enable per tenant extensions within shared schema.** Tenants need to add tenant-specific data extensions without affecting the overall data schema. Best practice designs include using a pre-allocated field in each table to hold custom XML data extensions or using name/value pair tables.
4. **Provide per tenant data backup and recovery.** SaaS customers are uneasy about entrusting all data and backups to a SaaS provider. Making per-tenant backup and recovery a part of the multi-tenant server framework eliminates these concerns.
5. **Integrate role-based user security with client-side UI.** There is often a disconnect between client-side and server-side security frameworks, opening vulnerabilities, particularly to malicious client code. The server-side role based access control framework should manage client access to UI widgets, services and data.
6. **Integrated deployment to cloud hosting environments.** Each cloud hosting provider has unique APIs and requirements. The multi-tenant server framework should include facilities to deploy easily to leading cloud providers such as Amazon EC2.
7. **Elastic allocation of cloud resources.** The cloud architecture should include the capability to dynamically expanding and contracting the pool of application and data servers.
8. **Transparent failure management.** When an application server fails, users should transparently failover. Similarly, when a database server fails, the application should failover to a backup server instance and restart a new backup server.

The Cloud Quick Start Architecture

IBM has worked with WaveMaker, RightScale and Amazon to define a cloud quick start architecture. Using this architecture, an ISV can greatly accelerate their delivery of SaaS and cloud products while minimizing costs and risks.

The following figure shows the cloud quick start architecture:



The elements of the cloud quick start architecture include

- **WaveMaker self-service client framework:** provides configurable, drag and drop client components that simplify the delivery of self-service web applications.
- **WaveMaker multi-tenant server framework:** provides secure, multi-tenant server modules that automate the creation of scalable, cloud-ready applications.
- **WebSphere:** provides scalable application middleware for on-demand business.
- **DB2:** provides cost effective data management for on-demand applications.
- **Amazon Elastic Compute Cloud:** provides highly scalable computing infrastructure
- **RightScale:** delivers demand-based scaling of the application infrastructure, transparent failover of application and database servers, automated data backup and automated restart of failed servers.

The following table shows how the cloud quick start architecture supports the best practices requirements for cloud computing.

Client-side requirements	Cloud quick start Solution	Provided By
1. Open, standards-based client technology	Client UI built using open-source Ajax framework	WaveMaker, Dojo
2. End-user personalization	User dashboard module includes personalization	WaveMaker
3. Internationalization and accessibility	Multi-language client UI supports Sec 508	WaveMaker, Dojo, WebSphere

4. Visual configuration of business workflows	Visual workflow module configures business rules	WaveMaker
5. Role-based user access to widgets & services	User management module includes RBAC	WaveMaker
6. Server-side data security	Client security module delegates to server	WaveMaker
7. Tenants customizable application templates	Client framework allows custom html, css templates	WaveMaker
8. Custom tenant widgets	Client framework supports custom, imported widgets	WaveMaker
Server-side requirements	Cloud quick start Solution	Provided By
1. Open, standards-based server framework	Server framework built using standard Java	WebSphere, WaveMaker
2. Secure tenant access to shared data	Server tenant module enforces tenant-filtered data views	WebSphere, WaveMaker
3. Per tenant extensions within shared schema	Server tenant module allows tenant-specific data for each table	DB2, WaveMaker
4. Tenant-specific data backup and recovery	Server tenant module allows per-tenant data backup and restore	DB2, WaveMaker
5. Server-driven security for client-side UI	Server security module integrated with client RBAC module	WebSphere, WaveMaker
6. Integrated deployment to cloud hosting provider	Server deployment module enables 1-click cloud deploy	Amazon, WaveMaker
7. Elastic allocation of cloud resources	Application and database servers scale with load	Amazon, RightScale
8. Transparent failure management	Application and database servers failover and recover automatically	RightScale

Open-source Platform As A Service (OPaaS) Requirements

[Platform as a Service \(PaaS\)](#) offers the potential to democratize web development by enabling anyone who can use a browser to assemble and extend web-based applications. Yet early PaaS players have introduced PaaS solutions that are remarkably proprietary, introducing high switching costs to move data or logic from one PaaS provider to another.

In contrast, an Open-source Platform as a Service (OPaaS) solution leverages industry standards and allows applications to be deployed across multiple cloud providers. An OPaaS solution has four characteristics:

1. **Open source** - the OPaaS solution is available as open source and supported by a large open source community.
2. **Portable** - developers can deploy OPaaS applications on multiple cloud infrastructures, including public and private clouds.
3. **Open server platform** - developers must be able to use standard languages and existing code within the OPaaS.
4. **Extensible client platform** - developers and end users must have standard, easy-to-use tools for configuring OPaaS user interface.

Next Steps: Planning Your Cloud Migration

ISVs who enter the cloud quick start program will be able to gain access to free consulting services to create a personalized cloud quick start Action plan. To join this program, you must be nominated by your IBM Technical Manager or apply directly to the cloud quick start program by sending an email to jnicke@us.ibm.com.

Nominations to the cloud quick start program will be evaluated based on the following criteria:

- Membership in IBM Partnerworld
- Usage of IBM products including WebSphere and DB2
- Timeframe for deploying a cloud-based solution

ISVs who are selected for this program receive the following services:

1. Free deep dive technical call (2-3 hours)
 - Understand As-Is & To-Be architecture
 - Review business drivers
 - Prepare tailored quick start roadmap
2. Free cloud proof of concept (1-2 days)
 - Build running cloud app using quick start architecture
 - Validate quick start roadmap recommendations
 - Prepare tailored quick start action plan

Additional Resources

1. IBM developerWorks article series: "Develop and Deploy Multi-Tenant Web-delivered Solutions using IBM middleware"
(link: http://www.ibm.com/developerworks/views/webservices/libraryview.jsp?search_by=Develop+and+Deploy+Multi-Tenant+Web-delivered+Solutions+using+IBM+middleware)
2. IBM SaaS Blueprints/IBM SaaS demonstration Series: These architectural patterns are described in published demos with sample code and [articles](#) showing how a multi-tenant application was built by exploiting existing features in multiple IBM middleware products. (link: http://www.ibm.com/isv/marketing/saas/demo_series.html)