TIER 3

6 Best Practices to Cloud Enable Your Apps

Expand Your Product Portfolio & Advance Your Software as a Service Offering

WHITE PAPER

vmware® vCLOUD® POWERED

CONTENTS

INTRODUCTION

Independent software vendors (ISVs) are increasingly turning to Software-as-a-Service (SaaS) as an alternative to traditional on-premise deployment. SaaS is hot! According to Gartner, 95% of organizations are planning to grow or maintain their SaaS investment. The influential technology blog GigaOm notes that the valuation of SaaS companies is skyrocketing compared to more traditional enterprise software vendors. If you currently offer your application on-premise, on dedicated physical servers, you are likely exploring the possibility of SaaS-enabling your product. And, if you're not, you probably should be.

Our goal with this paper is to share some proven best practices we have developed for SaaS enablement. There are a myriad of recommended practices for SaaS, but these six are likely the most relevant for your business. You can grab it and go! We have arrived at these practices after working with many ISVs on the process of taking an established on-premise application and transitioning it to a working, financially viable SaaS offering. The paper reviews technological and business objectives for SaaS enabling your application. Then, after setting context by contrasting SaaS with its ancestor, the application service provider (ASP) model, we go over the six most relevant technical best practices. These include verifying architecture, finding the right hosting environment, and setting up registration, management and billing, creating application templates and management practices such as performing data backups.

UNDERSTANDING YOUR OBJECTIVES FOR SAAS

SaaS-enabling your application will require an investment of time and resources. Such an undertaking should be done with clear objectives in view. Though everyone's goals for SaaS will be different, in general, a successful SaaS project aims to achieve an interlocking set of business and technological objectives. In business terms, SaaS is about enabling your clients – including future prospects – to take advantage of an Internet-based software delivery model. SaaS typically involves renting access to your application, in contrast to the traditional license-plus-maintenance model that most ISVs have used for years. The business advantage of SaaS is in its flexibility. SaaS makes it simpler and faster to on-board new clients. It can also be profitable to work with SaaS customers who are smaller than those that usually buy an on-premise version of your application. As a result, your business can scale faster and benefit from an expanded pool of clients. From a cost perspective, SaaS has the potential to drive increased profitability by streamlining infrastructure costs. However, this potential is not always realized by ISVs.

The technological objectives of SaaS should align with both revenue and cost management goals. The architectural decisions you make, the way you establish multi-tenancy and data, the hosting environment and so forth are all critical to ensuring that SaaS is a profitable path to growing your ISV business. For example, your ability to scale infrastructure efficiently will translate into smooth, low-cost on-boarding of new clients. Similarly, your ability to template applications will enable you to replicate your success with multiple versions of your application. Alternatively, automated monitoring and backups help you manage an expanding SaaS operation without the corresponding need to add support resources.

CONTEXT: APPLICATION HOSTING VS. SOFTWARE AS A SERVICE

What does it mean to "SaaS your app"? Making a conventional software application available on a SaaS basis is a deceptively simple process. You might be tempted to think of SaaS as just a new flavor of the old Application Server Provider (ASP) business model. It's not. One can become confused when so many products mindlessly stamp "cloud!" on their labels. In essence, ASPs operate software environments for customers while SaaS vendors rent their software in a scalable, self-service environment. To be fair, SaaS is an extension of the ideas introduced by ASPs, but there are fundamental differences between ASP and SaaS that affect their respective implementation practices. The following is a brief comparison between the two software delivery models.

	ASP	SaaS
Software Ownership	Software is either (1) owned by each customer and operated by ASP, or (2) owned by ASP with the environment provisioned for each customer.	Software created and maintained by SaaS vendor.
Software Architecture	Typically Web-based software built to scale for single customer.	Frequently built as series of decoupled components that cleanly scale for all customers.
Multi-tenancy	Each customer resides in its own environment, thus making maintenance time-consuming for ASP.	Each customer runs on shared infrastructure and often on the same software instance.
Infrastructure Abstraction	Customer sometimes has direct access to machines that run the software.	Customers only have access to the software itself.
Scalability	Favors a "scale up" strategy of providing more resources to existing hardware.	Built to scale up and out to accommodate user growth.
Cost Structure	Pay annual or long-term contract for hosting services.	Leverage per-user, pay-as-you go model where annual contracts are offered but not required.
Customization	Limited "free" customizations as each change increases support costs. Changes to the data model, business logic or security model are typically executed by the ASP.	Per-customer configurations and customizations that survive upgrades. Changes to data structure, business logic or security model are performed by the customer themselves.
Maintenance	Software upgrades or patches require environment downtime.	SaaS software often requires no scheduled maintenance times as rolling updates are performed.
Integration Strategy	Requires VPNs or scheduled, bulk data transfers.	Most SaaS products have published APIs for interfacing with the system.

THE SIX BEST PRACTICES

Keeping in mind your business and technological objectives, along with the ASP vs. SaaS context in mind, let's review our top six proven practices we have developed working with ISV clients over the last five years. Bear in mind that the core principle that SaaS should ideally be "self-service". SaaS should not require users to call up a service provider in order to implement any changes. By exposing supported, user-driven configuration in your architecture, you make self-service a reality and keep application support costs down. The following practices are far from exhaustive. However, they cover the major areas that we have found to be critical with success in SaaS enabling an application.

#1 Verify the Architecture

Before an application can be delivered in a SaaS style, it has to be evaluated for architectural readiness.

Stateless Web servers - In order to cleanly scale horizontally and provision new machines, it's essential that you build Web applications that don't maintain any local state. The Web servers should rely on some sort of shared database for their configuration. To offer a cloud-friendly application, the architecture must support no-touch elasticity. This cannot happen if you have Web servers with local state.

N**o hard-coded connections** - If your application has Web servers with hard coded values (e.g. IP addresses) for database connection strings or server-to-server communication, you'll have problems when you migrate your application to the cloud. You want to make sure that each individual layer of your application can scale independently without breaking connections between the tiers.

Extensible data model - This comes into play if you foresee subscriber-specific customizations taking place. Should customers be able to extend existing data objects, add new ones, or apply unique validation logic? If so, then it makes sense to design the data repository in such a way that customer extensions can take place.

Multi-tenant support - This isn't as straightforward as it may seem. One of the key principles associated with SaaS is putting multiple customers, or tenants, on a single server or software instance. The benefit of this model is that you gain operational efficiencies because you don't have to maintain each customer's environment uniquely. That said, multi-tenancy could happen at multiple layers of your application. Three viable options include:

- Provision unique Web applications and databases for each customer. While the underlying infrastructure may be shared among tenants, it is feasible to carve out unique application environments for each customer. The benefits include physical separation from other tenants (which may

be required in industries such as healthcare) and the option to upgrade each tenant on its own schedule. This starts to trend back towards an ASP model, but well-designed software coupled with a highly automated application provisioning process can still make this delivery model sustainable.

- Have customers share an application (version) but maintain their own unique databases. In this case, a single version of software is installed, but each customer configuration includes a reference to its own database. Here, physical data segmentation still exists and features such as per-customer encryption or direct database tunneling are possible, but overall application maintenance is simpler.
- Use a Salesforce.com-like model where all tenants share both an application version and database. Data is logical separated, but physical co-mingled.

Surfaced configurations - If someone expected to rent your software on an as-is basis, with absolutely no changes, then there is a limited need to expose user-driven configuration points. However, if you want customers to have the flexibility to extend the data model, change the application look-and-feel, define organization-specific workflows and set up security users/groups, all in a self-service fashion, then you have to design your software to support user-driven configuration changes.

APIs - If you aren't offering APIs, then you aren't offering a real cloud-based SaaS application. Applications without APIs become siloed and more difficult to integrate with or manage from afar. Good API design requires work, but the payoff is immense for the customers of the SaaS application.

Thoughtful security architecture - Security is a major consideration when building an application that may be shared by many diverse users. Security issues that arise include how you identify (authenticate) and assign permissions (authorize) to users, how you encrypt data at rest, how you secure data in transmit, provide audit trails and so forth. Ideally, your application allows Single Sign On (SSO) through a standard mechanism such as Security Assertion Markup Language (SAML) which then requires one fewer password for users to create and maintain.

Integration with other cloud platforms - While not a necessity, it's powerful when cloud applications embrace other cloud applications. Users of your SaaS application might benefit from integration with Google Docs or Microsoft's SharePoint Online, for example.

#2 Find the Right Hosting Environment

Creating SaaS-friendly application architecture is a great first step to business success in the cloud. Next, you have to deploy it in a way that maximizes reliability while minimizing support costs. Even a well-architected application will flounder if it is stuck in a sub-par hosting environment. You will want to find a mature Infrastructure-as-a-Service (IaaS) provider with a mature, elastic environment that has strong capabilities in the following areas:

Metered Billing - Pay-as-you-go pricing is a major attraction of SaaS for clients. You want to make sure that you can easily capture usage metrics that can factor into the monthly cost. While you may simply charge a flat price per application user (and forego chargebacks for individual infrastructure costs such as storage, bandwidth and CPU cycles), it will be useful for your hosting provider to show you a clean breakdown of the costs associated with your SaaS application.

Quick horizontal/vertical scale - Increasing RAM or CPU sometimes makes sense when you need more horsepower for a given application server. But recall that "cloud" is associated with easy horizontal scale of commodity hardware. The platform beneath your SaaS application needs to be able to scale automatically and rapidly through a self-service interface.

Access to Web-based installers - Instead of leveraging virtual machine (VM) snapshots and rehydrating them when you need more servers, consider building the servers automatically when scaling is required. If you use VM snapshots, you still have to deal with patching and managing them so that they are usable when you need them. Instead, if you have access to the application source code (or Web installers for packaged software), you can use any number of tools to quickly and reliably build new servers.

Active monitoring - In order to be able to support a large number of customers on your application, you'll need to be able to respond quickly to the inevitable issues that arise with the hardware or software. Your SaaS hosting platform needs to be watching the health of its servers vigorously. The platform needs to be able to notify you of problems and perform prescribed responses (e.g. reboot server, take offending server offline, add more servers).

Global deployment options - One value of SaaS software is that it's on the public Internet. This means that the application may be accessed by users across the globe. If there's a chance that your SaaS application would be used by global users, consider hosting providers that have an international presence and support provisioning across global data centers.

One-click provisioning - A cloud application architecture may not be simple. Your software may have numerous front end Web servers, a Web service layer, distributed database system, batch job processing servers and more. If possible, try and find an Infrastructure-as-a-Service (IaaS) provider that allows for "templating" a solution stack and supports a single-click deployment option.

Robust backup and restore options - Disasters happen. Even the best cloud environment experiences unexpected issues that can take down an entire data center. You want to make sure that your application data can be readily (and regularly) persisted in a location outside the primary data center. Disaster recovery planning is serious business and ideally, your hosting partner has lots of experience in this arena and can provide both the thought leadership and tools that make this a reliable capability.

#3 Provide Registration, Management and Billing Services

If the application is well built, and hosted on world-class infrastructure, all that's left is to make it simple for customers to consume it. How will you make it easy for customers to evaluate, buy and get started quickly with your application? In order to offer a SaaS product that requires minimal manual intervention, you should address the following areas:

Sign-up pages - This may seem obvious, but you will want to construct a straightforward way for customers to get started quickly with your software. If your sign-up process requires someone to call a phone number, that is not optimal. It should be a completely browser-based experience.

Management dashboard – There is a strong business value in surfacing application configuration details to allow the customer to make certain changes to the appearance or functionality of the application. Your customer shouldn't have to make these changes by executing a series of shell scripts or REST API calls. Rather, they must have an effortless way to view and edit configurable values. Likewise, a good management dashboard also exposes key capabilities such as security role definitions (and user provisioning) and billing services.

Data import/export services – Think about how new clients will migrate old data onto their new instance of your SaaS application? If your answer is "key it in by hand," then you have started the business relationship with a hassle for the client, one that might be an obstacle to adoption. Data import tools make it possible to start using an application right away, and, create some instant data gravity along the way. Just as important as supporting easy data import capabilities, you need to offer a way to pull data out of your application. While the natural inclination may be to lock people into your platform, you're both harming your customer and making simple integration scenarios harder.

#4 Create Application Templates

Unless your SaaS offering is uniform and low-priced, your setup will be optimal if you let each customer have its own Web application and database instance but potentially share underlying infrastructure. Each customer may want to perform some unique customizations that wouldn't be shared by all application users. It can be a challenge, however, to remain efficient while being so flexible about provisioning. For example, your application might consist of a database server, fronted by two web servers, as shown in Figure one. In this case, this configuration would be a "building block" for your SaaS offering. The question becomes, how can you deploy multiple blocks with as little friction as possible? Excessive manual setup can rob your SaaS initiative of profitability.

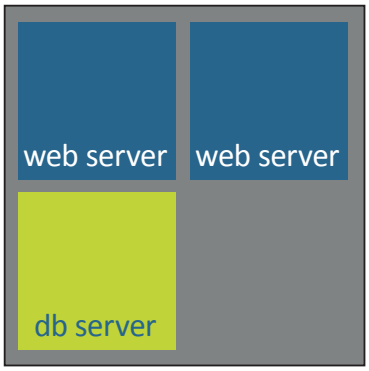

Figure 1 – Standard “two tier” application architecture as an example of a building block for a SaaS deployment.

You can only do this at scale if your SaaS platform has significant automation capabilities that can handle server scale-out with limited manual intervention.

Some vendors, Tier 3 included, solve this challenge by enabling you to create templates for your application and deploy by installing servers in groups. With these tools, you can create a template for your application building blocks and replicate them as you scale your SaaS deployments. These platforms let you package a Web application for SaaS provisioning, with environment automation capabilities that make management of distinct “per customer” environments entirely manageable. Servers are managed in groups so each customer will be provisioned into their own group. This makes billing, scale thresholds, and configuration much more personalized. You can also use scripts to expedite and automate many maintenance tasks such as OS upgrades and patching of templated groups of application building blocks.

Let’s consider how one type of cloud orchestration tool can help automate environment build outs. Using Tier 3’s template technology, you can turn a server it into a reusable image. The Tier 3 platform refers to this as a “Blueprint”. It’s easy to model standalone SaaS instances. Figure 2 provides a simple reference architecture for how Tier 3 brings together multiple two-tier client “building block” instances with a repository of application template blueprints and a build engine that automatically creates new instances based on customer configuration needs. Blueprints can be rapidly deployed by the Tier 3 build engine to any of our global datacenters.

p9

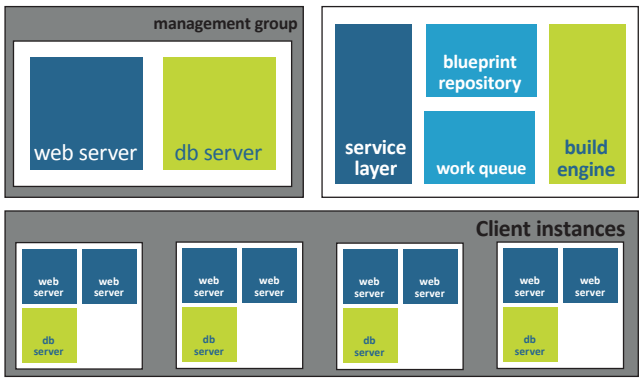

Figure 2 – A standardized approach to replicating applications using templates and a “build engine”.

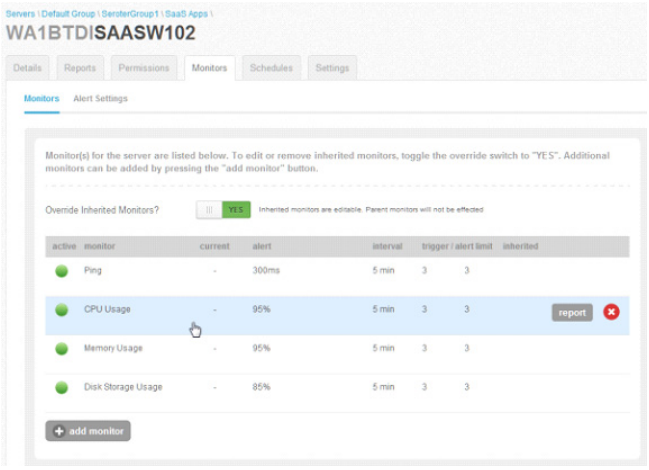

Figure 3 – The Tier 3 Control Portal, allowing administrators to set thresholds for alerts on application performance.

#5 Monitor Performance and Scale Environments

While everyone hopes to provision servers that can withstand whatever load gets sent its way, there often comes a time to reassess the resources that have been assigned to the application. Some cloud platforms offer a range of performance monitors that paint a picture of a server's health.

Consider a scenario in which the Web servers should not exceed 90% CPU for a long duration. Using an online monitoring tool from the cloud management portal (Tier 3's is shown below) our SaaS administrator can view the monitors for an individual Web server and override the inherited CPU "95% threshold" that come from the group.

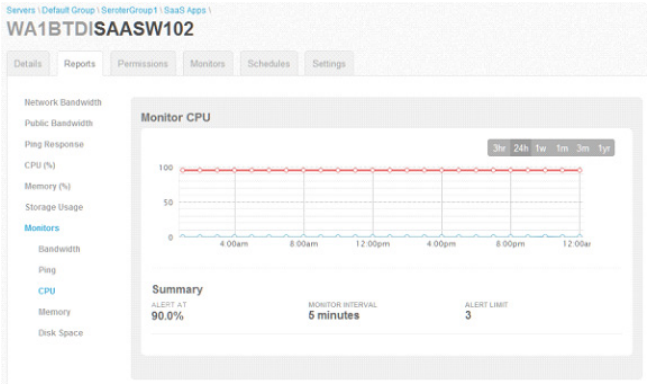

Figure 4 – An example of an online monitoring capability.

The administrator can then change the alert threshold to 90%, and, define which user in the account will receive the email notification when the threshold is exceeded for an extended duration. While alert messages are used for proactive notification, an administrator can also view reports that track usage compared to monitor thresholds.

If the SaaS administrator observes a sustained spike in usage and needs to expand the application footprint, he or she has multiple options. An administrator has the option of scaling horizontally by adding new servers. This can be done via the typical server provisioning process or using cloud orchestration functions. The other option involves scaling the server vertically by expanding the server's available resources. For some operating systems, such as Windows Server 2008 R2, this capacity adjustment can occur without taking the server offline.

#6 Perform Data Backups

One of the biggest fears of an application owner is experiencing a major crash and realizing that critical data is lost and unrecoverable. In the scenario which each customer has its own environment, the need for a consistent and comprehensive backup strategy is even more imperative. A few cloud providers perform automatic daily backups. Make sure that each backup contains the full state and data of the server. For the most demanding customers, a rolling fourteen day backup could be in order. All of this ensures that customers can rest easy knowing that they only face a minimal data loss in the case of a catastrophic event.

For many modern Web applications, the applications servers themselves don't maintain much (any?) state and can be added and taken offline with little impact. The real heart of most applications is the data repository. If a your customer wants to do database-level backups, they could choose to manually log into the database server and backup the critical data. However, that is extremely time consuming and inefficient. You should choose a cloud hosting provider that supports custom database backups via APIs or created by network operations experts.

CONCLUSION

As you can see, SaaS-enabling your application involves much more than just taking an existing software package and pushing it out to a few cloud servers. In this review of six best practices, we hope we've shown you the importance of architecting, automating, managing and supporting your software in way that makes it both efficient and profitable. Each practice is required for success on its own, though realistically, they all have to be approached as a group to get the true benefit of SaaS in your ISV business.