Everything you need to know about SaaS

The Complete Guide to Software as a Service

Robert Michon

The Complete Guide to

Software as a Service

ROBERT MICHON

The Complete Guide to Software as a Service
Everything you need to know about SaaS

First Edition (revision 2) – August 2017

The web addresses referenced in this book were live and accurate at the time of publication but may be subject to change.

Both Sides, Now – lyrics by Joni Mitchell; © 1967 Gandalf Publishing
Used with permission from Alfred Music, Inc.

In keeping with the philosophy of the Pledge 1% movement, one percent of all proceeds of this book will be donated to a charity.
For more information, consult www.pledge1percent.org

For information, comments, corrections and suggestions contact:
CompleteGuideToSaaS@gmail.com

Cover image by ShutterStock

ISBN: 978-1546308492

Contents

PREFACE

In 2003, I left a successful career in consulting to join a four-year old software company, building a recruiting and Human Capital Management (HCM) solution and delivering it over the Internet. Our company was about the same age as an upstart tech company in San Francisco called Salesforce who was calling for "the end of software".

At that time, my company was aggressively signing new clients ("acquiring logos" as it were) to gain relevant market share. I was running IT operations and was faced with several challenges as the business grew 100% every year.

The first challenge was scalability. We hosted our solution in an off-site colocation facility and had to plan all expansions to our hardware. Install too little hardware and you miss a sale. Install too much and you have the CFO heading down to your office. In addition to technical scalability (our databases were doubling in size every 160 days), we also had to find a way to scale operationally. If we doubled the number of clients, we could not expect to double the number of operations staff. Process optimization and automation were imperative.

The second challenge was retaining our existing clients. Most clients signed a three-year contract and many of our larger clients were coming up for renewal just as I was coming onboard. Even though management's focus was sell, sell, sell; our renewing clients were demanding service, service, service. After three years, our clients had become much more knowledgeable about SaaS, what demands to make of their SaaS provider and what language they expected in their contracts.

So, my team had to rapidly come up with credible answers to questions on network security, data segregation, disaster recovery, uptime guarantees, maintenance windows, etc.

There were many other challenges about data sovereignty and the choice of data center locations, contracts with hardware suppliers, follow-the-sun support models, zero downtime software maintenance, etc. The list goes on and on.

After leaving this SaaS provider around the time of their IPO, I went on to help other companies on the same journey. I have encountered SaaS providers that were basically running their business on Excel spreadsheets and were about to implode. Others were transitioning from an on-premise licensing model to SaaS, where I heard Support staff asking their SaaS clients to “send in a dump” so that they could reproduce the problem. Others had customized their software extensively for each client such that they were running over two dozen code versions in production with no hope to achieve any significant scalability or automation.

In my mind, the basic tenants of a successful SaaS company were clear. Yet, I realized that what I considered obvious was not always what was being implemented in the real world. The elements of success of a SaaS company had not been compiled in a single work. There were some excellent blogs that discuss particular elements of the equation, but nothing tied it all together. That is why I wrote this book.

This book is a must read for entrepreneurs who are launching a SaaS company. It will also guide any software company who is transitioning from an on-premise license model to SaaS. It also provides useful information and insight to different functional managers within a SaaS company. My hope is that users of SaaS software will also find the contents interesting as they become more knowledgeable clients of their SaaS providers.

The contents of this book cover the definition and the origins of SaaS. It then goes on to describe the functional, operational, technical and financial aspects of SaaS. The final chapters discuss security, sales and contracting.

I wrote this book to fill a void, as it exists today, in the world of SaaS. Yet, software paradigms continue to evolve as we move to micro-services and an API-driven software economy. So, the definitions and premises of this book may evolve to something new in the future. I am glad to have taken one step of that journey with you, through this book.

ACKNOWLEDGEMENTS

First and foremost, I wish to thank my wife, Elaine, for putting up my late nights and weekends staring into a computer screen. Writing this book has consumed many hours that were only possible to do with her full support and encouragement.

A special thanks to Martin, my boss at Taleo, my associate at SaaS360 and ski companion for the last twenty-five years. Thanks for introducing me to the world of SaaS. Thanks for the many discussions, comments and criticisms as the content of this book took shape.

Thanks to Kai Wang, who gave Martin and I full confidence in reshaping his SaaS organization. His support allowed us to experiment and perfect many of the concepts presented here.

Kudos as well to my SaaS Operations Team at Taleo: Martin T., André L., Bruno S., and the B brothers: Phil & Pat. You showed me the true meaning of dedication, innovation and team-work. It was a great ride together.

As a continual learner, I would like to thank the many teachers, mentors and coaches that I have had the pleasure of meeting over the years. You have, probably unknowingly, contributed significantly to what I have become today. I will try to pass it forward as best I can.

Dedicated to brain tumor survivors everywhere

1. INTRODUCTION

I've looked at clouds from both sides now
From up and down and still somehow
It's cloud's illusions I recall
I really don't know clouds at all
- *Joni Mitchell*

Cloud computing is widely used, but often misunderstood term. When people hear that their data is "in the cloud", they often imagine an amorphous, almost magical, computing and storage environment. Others associate *cloud computing* with a specific service offering such as Amazon Web Services (AWS), Google Cloud Platform, IBM Softlayer and Microsoft Azure.

In this **Introduction**, we will define cloud computing and explain its three main variants: Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS). We will also explain the differences between a public cloud, private cloud and hybrid cloud. In short, we will learn what Paul Maitz, CEO of VMWare, meant when he said: "Cloud is about how you do computing, not where you do computing."

1.1 What is the Cloud?

In fact, “the cloud” is not amorphous but a real and physical, but ever-changing, collection of servers and storage units, located in data centers and tied together through fiber optic networks routing through peering points that are accessible through your Internet Service Provider at your home or office, or accessible through your cellular network provider and your smart phone.

One of the key characteristics of the Cloud is the **appearance of infinite compute and storage resources**. When you access a cloud provider, such as Amazon Web Services (AWS), you can spin up any number of virtual machines (VMs) and tie that to as many terabytes of storage that you wish. From the Cloud provider’s perspective, they must have only one more VM available than the greatest number of VMs ordered at any one time to create the illusion of infinite compute capacity. Likewise, if there is at least one unreserved terabyte of storage in the Cloud provider’s facility, the consumer retains his illusion of infinite storage available to him.

Another key characteristic of the Cloud is **rapid scalability**. The Netflix use case is often quoted in this context. Netflix will double its streaming capability just before the weekend by scaling up its AWS infrastructure on Friday. It will then “turn off” excess capacity on Monday to the level required for the work week when there is less demand for its streaming service.

Another example of scalability is in system testing and prototyping. Spinning up multiple test environments, in the cloud, just before a major release and then releasing those resources at the end of the QA cycle reduces testing costs and increases software quality by allowing a more complete testing environment. Indeed, many DevOps implementations[1] are based on this agility in creating multiple test environments.

Finally, converting infrastructure costs from a large capital expense, up front, to a **pay-as-you-go** operational cost has enabled many startups to get off the ground while reducing the amount of cash required to set up an initial operational infrastructure. Companies adopting a cloud computing service trade upfront capital costs (CAPEX) to recurring operational costs (OPEX).

Instagram, at the time, were purchased by Facebook in 2012 for $1 billion. Instagram was just 13 people with over 100 servers running in AWS, so little capital outlay and few physical assets. Indeed, Instagram had not purchased any servers or storage to run their business: it was all paid for a monthly fee to AWS. The real asset that Instagram had accumulated was their 30 million users.

In summary, cloud computing is an alternative to installing and maintaining your own physical computing infrastructure resources. There are different service models and different deployment models, each one meeting a different set of requirements and constraints.

We will present these service models and deployment models in the following sections.

1.2 The Cloud Model

The National Institute of Standards and Technology (NIST) offers this definition of the Cloud:

> "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. This cloud model is composed of five essential characteristics**,** three service models, and four deployment models." (1)

The three service models: IaaS, PaaS & SaaS, all share the same five essential characteristics of cloud computing:

1. **on-demand self-service**; most cloud services will provide administration screens where the end user can add or remove services themselves without the intervention of a technical support person;
2. **broad network access**, usually through the Internet but also through dedicated fiber links (aka "direct connect");
3. **resource pooling**, through virtualization and partitioning techniques;
4. **rapid elasticity**, this addition or removal of computing resources can be done dynamically or in a matter of seconds or minutes, with the illusion of infinite computing resources;
5. **measured service**, with a pay-as-you-go model.

To describe quite simply the differences between the three service models, we can say that:

- Software as a Service (SaaS) is a specific application that is designed for end-users to do useful work, delivered over the web.
- Platform as a Service (PaaS) is the set of tools and services designed to make coding and deploying those applications quick and efficient;

- Infrastructure as a Service (IaaS) is the hardware and software that underlies all computer processing and storage – servers, SANs, networks, operating systems.

However, all three service models are “as a Service”. They all present a defined and usable service to a consumer, whether that consumer is an application user, a software developer, or an IT systems administrator.

The transport analogy

An often-used analogy to describe the difference between Infrastructure, Platform and Software is the highway system. The Interstate highway system, consisting of roads, bridges and tunnels, is **infrastructure**. By itself, it produces no outcomes, but enables the other activities. The cars, trucks and buses that run on the highways are **platforms**. They also exist to enable the actual economic activity of transporting people and goods from point A to point B. The actual transportation of people and goods corresponds to the software layer since that is where actual business activities happen.

The three service models, IaaS, PaaS & SaaS, each present a different level of abstraction from the actual physical hardware and differ by what you control and what you manage. The main difference between these three models can be illustrated in the depth of services provided in what is commonly called the Cloud Stack. This is shown in the following diagram.

As can be expected, the Tenant of a SaaS service will have delegated many activities to the SaaS Service Provider. At the other end of the spectrum, the IaaS Tenant delegates a much more limited set of responsibilities and retains the remainder for himself.

In a traditional computing environment, bringing online a working system involves procuring, installing, configuring all the services in the cloud stack. This would involve several technical selection exercises, prototypes, or proof of concepts for each of the core technologies that comprise the solution. Then license agreements and hardware financing agreements must be negotiated. Then, besides building the actual application, the IT team must put in place management & monitoring, implement and test fail-over and disaster recovery mechanisms, harden the environment against intrusion, etc. And the IT Team would also be required to build out a data center facility or select a colocation provider.

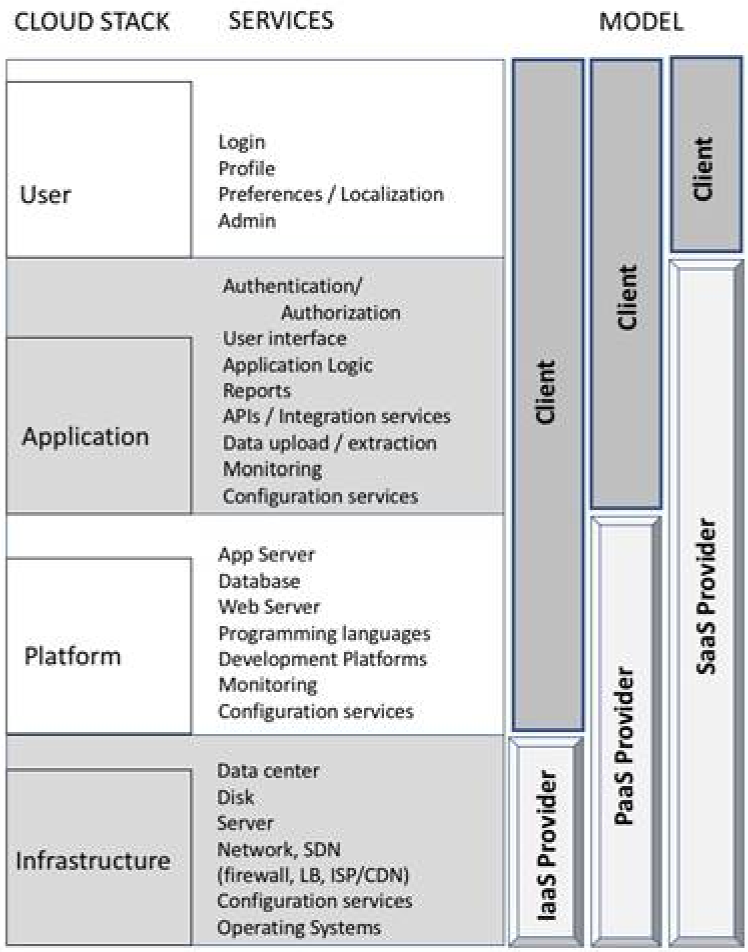


figure 1 – Three Service Models of Cloud Computing

In today’s application driven economy, time to market is critical. Limiting the capital required to achieve product-market fit and rapidly gain market share is a key success factor. Both factors, time to market and limiting capital requirements, can be leveraged by using a cloud solution, be it IaaS, PaaS or SaaS. By obtaining certain portions of the Cloud Stack as a service, businesses can concentrate themselves on their unique value-creating activities.

As illustrated in figure 2, the lower initial purchase cost of On Premise license software hides many installation & ongoing activities that need to be factored-in to get a valid comparison with a SaaS solution.

The choice of IaaS, PaaS or SaaS is predicated on the type and characteristics of the application that you want to bring online and the “fit” of the IaaS, PaaS or SaaS solution to what you are trying to build.

To further complicate the procurement process, some Cloud providers will offer both an IaaS solution and an PaaS solution under the same marketing brand. And increasingly, users of these services will leverage both the IaaS and the PaaS services for different requirements.

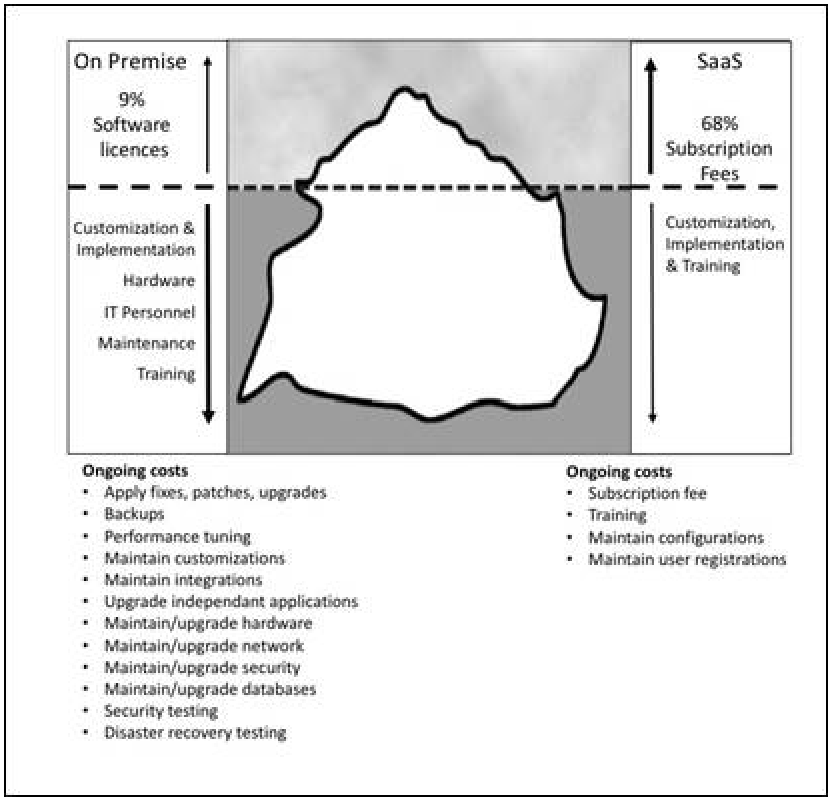
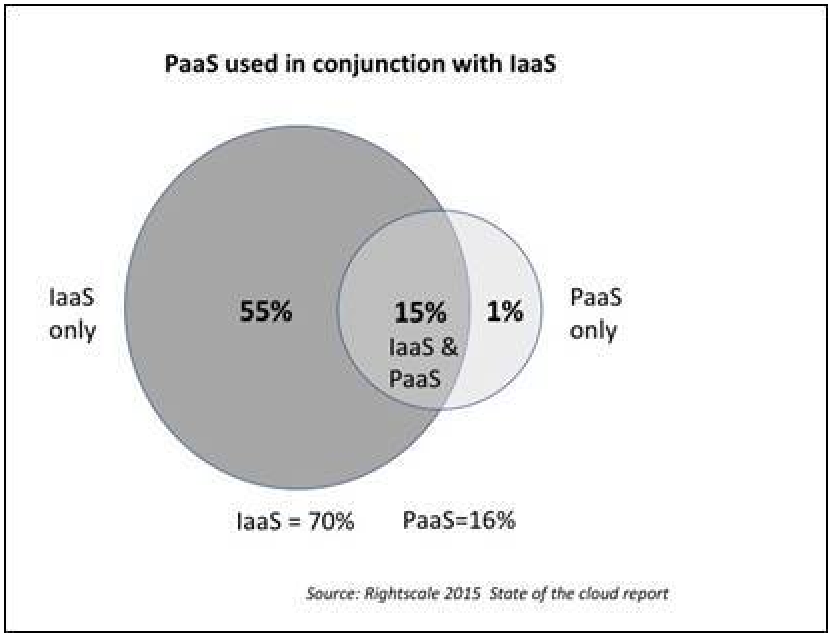


figure 2 – Ongoing costs of On Premise vs SaaS

From the 2015 RightScale ***State of the Cloud Report*** (2) we can observe that IaaS solutions are more widely used and that the clear majority of PaaS users also use an IaaS solution simultaneously (see figure 3). And indeed, the distinction between the two service models can sometimes get blurry.

The same "overlap" can exist between the SaaS and PaaS solutions of the same provider. A well-known example is Salesforce, a leading SaaS supplier of CRM and salesforce automation solutions. Salesforce also makes available, as a PaaS solution, the same development platform that it uses internally as a PaaS solution. **Force.com** is a Platform as a Service (PaaS) that allows developers to create multitenant add-on applications that can also integrate into the main Salesforce.com application. **Force.com** applications are hosted on Salesforce.com's infrastructure.



Source: Rightscale 2015 State of the cloud report

figure 3 – IaaS and PaaS Combined Usage

1.3 IaaS

To define in more detail the IaaS service model, we start again with the standardized NIST definition:

> Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision processing, storage, networks, and other fundamental computing resources where the consumer is able to deploy and run arbitrary software, which can include operating systems and applications. The consumer does not manage or control the underlying cloud infrastructure but has control over operating systems, storage, and deployed applications; and possibly limited control of select networking components (e.g., host firewalls). (3)

There are many IaaS suppliers, some notable ones being Amazon Web Services (AWS), Google Cloud Platform and IBM Softlayer. There are many other providers, including some regional offerings.

These IaaS offerings meet the five characteristics of cloud computing that we have defined before. Most IaaS providers will allow you to select and configure infrastructure elements with a simple "point and click" management console that can be manipulated by someone with a skill level that is much lower than that required to build an equal infrastructure "from scratch".

An IaaS solution is particularly well suited to the following situations:

- A start-up that does not have a lot of capital to invest in hardware and software infrastructure. As we will see later, a startup should be conserving cash to invest more in its product development and its sales efforts, not buying hardware;
- Companies that are growing rapidly; scaling the infrastructure to match that growth can be challenge, both financially and logistically;
- Solutions where demand has important fluctuations. Building out an infrastructure to meet all the peaks could leave significant unutilized resources when demand is in a trough;

- For specific lines of business, trials or temporary needs. For example, President Obama’s first election bid utilized significant computing resources that were no longer needed after the election. The web site for the pre-launch and hype period of a major motion picture is another example.

Yet, an IaaS solution might not be such a good option in the following situations:

- Where regulatory or legal compliance imposes certain restrictions or limitations on where data may be stored, that cannot be guaranteed by the IaaS supplier;
- Where the application has significant requirements for high levels of performance that can be best met with on-premise or dedicated infrastructure;
- Any specific configuration or communication requirement that precludes an IaaS solution. As an example, I assisted a software company to port their on-premise solution to a SaaS solution. But their solution utilized multicasting as a communication mechanism between servers; this functionality was not supported at the time by the IaaS vendors, forcing us to choose an in-house hosting solution.

1.4 PaaS

Not to be confused with PAAS, which is a registered trademark of Signature Brands LLC for a brand of Easter egg dye.

Similarly, here is the NIST definition of PaaS:

> Platform as a Service (PaaS). The capability provided to the consumer is to deploy onto the cloud infrastructure consumer-created or acquired applications created using programming languages, libraries, services, and tools supported by the provider. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, or storage, but has control over the deployed applications and possibly configuration settings for the application-hosting environment. (3)

Although NIST uses the word "consumer" in its definition, the actual target audience for PaaS is typically software developers. PaaS can be considered as a computing platform that allows the creation of web applications quickly and easily and without the complexity of buying and maintaining the software and infrastructure underneath it. In other words, rather than being software delivered over the web, it is a platform for the creation of software, delivered over the web.

Although there are many different "flavors" of PaaS, some on the common characteristics are:

- Services to develop, test, deploy, host and manage applications in an integrated development environment. The PaaS solution should include all services required to support a complete software development lifecycle.
- User Interface creation tools that allow the creation, modification, testing and deployment of user interface layer of the application;
- Ability to scale the processing power that supports the deployed software to meet variations in workloads;
- Integration with web services and database services.

Depending on the service, we might also find tools to assist in team collaboration between members of the development team, including project planning and tracking. In some cases, we might find services to assist in managing client subscriptions and billing. Almost invariably, the PaaS services will be supported by a multi-tenant architecture where many concurrent users leverage the same PaaS infrastructure in virtual isolation.

As mentioned before, Salesforce offers Force.com as a PaaS solution, which is essentially the same platform that they use to run their SaaS offering. Choosing Force.com is an obvious choice for a software developer who wants his solution to leverage data contained within the Salesforce CRM solution and/or wants to distribute his solution through Saleforce's App Exchange.

At another end of the spectrum, Heroku is a PaaS solution which has a greater independence for programming languages and databases. Originally released with only the Ruby programming language, it now supports many languages such as java, node.js, python, PHP, etc. In addition to the PostgreSQL database, it now also supports the MongoDB and Redis databases.

Other examples of PaaS services include the Google App Engine and Microsoft Azure Services.

The use of a PaaS solution can be especially powerful where there are many developers, co-located or not, and where the development process, such as Agile, predicates frequent interactions with external parties, such as the user representatives. The availability of automated testing and deployment services can increase development productivity and quality significantly. Likewise, the availability of libraries of pre-defined components, frameworks and building blocks can speed up the development process such that a PaaS approaches will be more and more the norm.

However, PaaS may not be a judicious choice where application portability is a must. Indeed, for most PaaS solutions, there is an implicit form of vendor lock-in either from the hosting infrastructure or proprietary languages or frameworks. For example, an application that was developed on the Force.com platform will not migrate transparently to Microsoft Azure or to a self-hosted platform.

Another context where PaaS may not be the best choice is where specific application performance objectives need that the underlying hardware or software must be customized in some fashion.

1.5 SaaS

The definition of Software as a Service, once again drawing from NIST:

> *Software as a Service (SaaS).* The capability provided to the consumer is to use the provider's applications running on a cloud infrastructure. The applications are accessible from various client devices through either a thin client interface, such as a web browser (e.g., web-based email), or a program interface. The consumer does not manage or control the underlying cloud infrastructure including network, servers, operating systems, storage, or even individual application capabilities, with the possible exception of limited user-specific application configuration settings. (3)

We find many of the characteristics of cloud computing embedded in the definition of SaaS. Notably:

- Access through a network;
- Resource pooling, with little end-user control of the underlying infrastructure;
- An on-demand, self-service model.

As alluded to in this definition, SaaS makes sense where the "vanilla" version of an application meets the needs of the tenant organizations. A good example of this is an email service, where most requirements can be met with a standard product.

SaaS also makes sense when there are extensive mobility requirements or significant interactions with third-party stakeholders. Using a SaaS solution in these cases means that the user organization does not have to build an extensive content distribution network; it can leverage the SaaS provider's network. Likewise, in the case of extensive third-party access, it is the SaaS provider's task to ensure adequate network access controls and the user organization does not have to grant outside access to its internally hosted systems.

But SaaS may not be the preferred option in cases where:

- Extensive personalization of the application is required;
- Where regulatory or legal compliance imposes certain restrictions or limitations on where data may be stored, that cannot be guaranteed by the SaaS supplier;
- Where real-time integrations are required with up-stream and down-stream systems; this can become difficult to monitor and recovery from system failures extremely difficult to orchestrate.

For this reason, SaaS solutions are popular for non-core business functions that can operate in a more-or-less stand-alone fashion, for example, CRM, Recruiting and Human Capital Management, payroll, etc.

In Summary

How to choose the right cloud service model: IaaS, PaaS or SaaS? In summary, an IaaS solution makes sense for a customer who has a specific software application to develop or a software package that he wants to install and run himself. However, if the software to be developed or deployed does not rely on any specific hardware or operating system, then a PaaS solution may save time and complexity. Finally, if a business need can be met with an existing service that already meets the clear majority of a client's business requirements, then a subscription to a SaaS service will allow the client to be fully functional in the shortest time possible, with the lowest capital expenditure.

A Cloud Advocacy group summarizes the benefits of the cloud computing model in five points (4):

- Achieve economies of scale;
- Reduce CAPEX by moving to OPEX;
- Improve access;
- Implement agile development at low cost;
- Leverage global workforce.

Simply put, cloud computing enables both IT agility and business agility, at a lower cost.

1.6 Deployment models

As referenced in the NIST definition (1), there are four deployment models for cloud computing. These deployment models are:

- **Private Cloud.** In this deployment model, the cloud infrastructure is provisioned for use by a single organization, made up of many consumers, for example business units. This is a typical scenario where a large corporation or government places IT in Shared Services organization. The private cloud may be hosted and managed by the organization itself, or a third party. The private cloud infrastructure may be hosted on premise or off premise.

  Since the private cloud is used by a single organization, this attenuates some of the data privacy concerns that arise in a multi-tenant environment, especially in public clouds. As well, the organization has more control on the actual underlying infrastructure. But the economies of scale and rapid elasticity may be constrained.

- **Public Cloud**. This cloud infrastructure is provisioned for open use by the general public. The well-known cloud services such as Salesforce, Netsuite, Amazon Web Services, Microsoft Azure and the like are all public clouds.

  It is in this public cloud scenario that consumers of the service can leverage the biggest economies of scale and the best utility pricing. However, they achieve this with a loss of control over:

  - Infrastructure configurations; the public cloud provider may offer a limited number of configurations options.
  - Data placement; at best, the consumer may be able to specify a “computing region”;
  - Service Levels and performance: the consumer is dependent on the cloud provider to meet these requirements;
  - Backups and Disaster recovery: although the consumer can design these elements with on an IaaS service, the control shifts

towards the Provider of PaaS and SaaS services.

- **Community Cloud**. This cloud deployment model services a community of consumers from organizations that have shared interest (mission, security requirements, policy, etc.). It may be hosted and managed by a member of the community or by a third-party.

  The community cloud model allows for more control than the public cloud model. This may be of importance where data privacy is a concern. For example, a community cloud in the health care sector could provide common applications to a wide variety of healthcare providers in a specific geography.

- **Hybrid Cloud**. A hybrid cloud is simply a combination of two or more distinct cloud infrastructures (private, public, community) bound together to achieve data or processing portability between the elements.

  For example, an organization could use its private cloud where data privacy concerns are great, but leverage public cloud resources when rapid elasticity and resource pooling are required.

Although not specifically mentioned in the NIST list of four deployment models, there is the concept of **virtual private cloud** that is worth defining. A virtual private cloud (VPC) is a pool of resources, allocated within a public cloud infrastructure, but with sufficient isolation from other users of the public cloud such that it mimics the characteristics of a private cloud infrastructure. As we will see in Section 5.4, AWS offers a VPC service as one of the recommended ways of implementing a multi-tenant environment in AWS.

More information on the advantages and disadvantages of each deployment model can be found in NIST Publication 800-146: *NIST Cloud Computing Synopsis and Recommendations*.

2. THE ORIGINS OF SAAS

The original SaaS: Smoked meat as a Sandwich
Schwartz's Delicatessen in Montréal.

SaaS is one of the three service models of cloud computing. How did the software industry evolve towards this model? Some analysts retort that SaaS isn't anything new and that the industry has been provided time sharing and managed services for decades. These analysts don't consider that SaaS is anything new or innovative.

In this section, we will explore the origins of SaaS. We will review its key characteristics and identify what makes it a unique software delivery model.

2.1 Mainframes and packaged software

In the 60s and 70s, the computer industry was dominated by the mainframe manufacturers, collectively known as IBM and the Seven Dwarfs[2]. Named as such since IBM's market share surpassed the market share of its seven closest competitors put together. At that time, software was sold by, and often installed by, the hardware manufacturer.

This strategy of "bundling" hardware and software attracted the attention of US antitrust regulators, who sued IBM for improper "tying" in 1969, alleging that it was an antitrust violation that customers who wanted to obtain IBM software had to also buy or lease IBM hardware in order to do so[3]. Although the case was eventually dropped by the US Justice Department, after many years of attrition, as "without merit", IBM and others separated their hardware and software, allowing third parties to write and sell software to users of another company's hardware.

This "unbundling" allowed the growth of a dynamic software industry serving both the mainframe and the newer mini-computer market. The most popular applications for packaged software targeted core business systems; payroll, financial systems and operational processes.

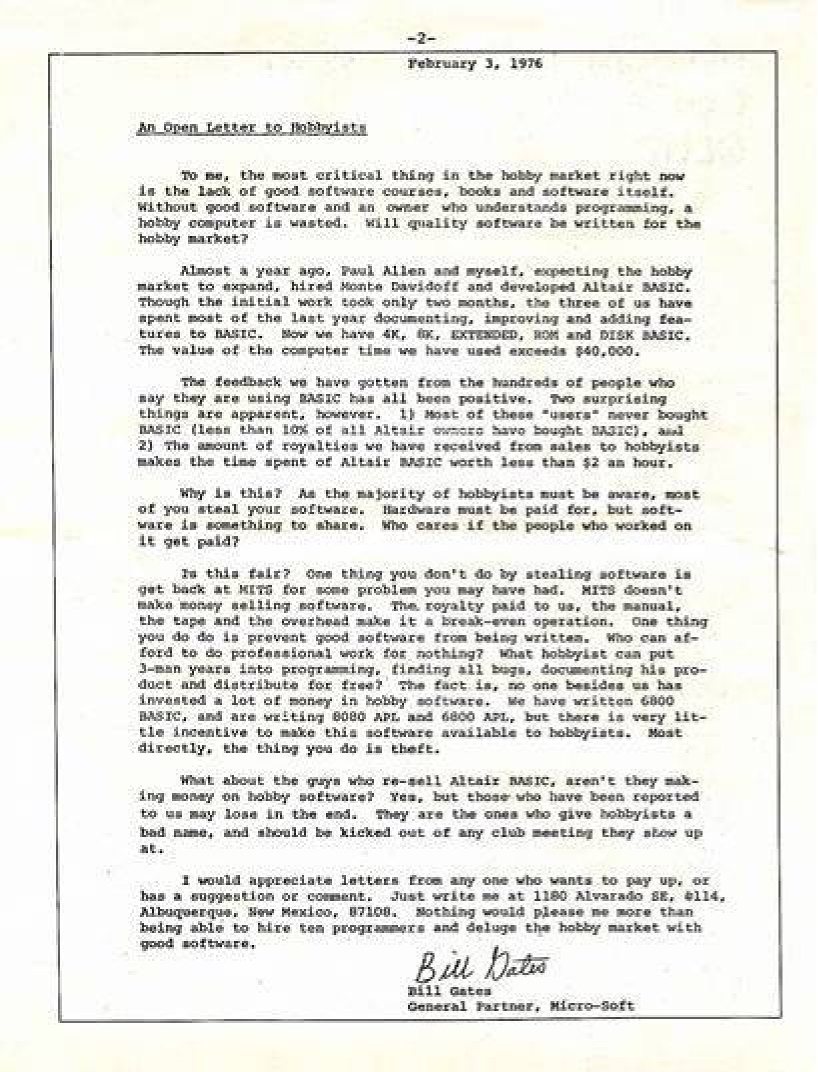
1972 saw the creation of SAP in Germany and in 1977 Oracle was founded in a nascent Silicon Valley. SAP and Oracle would become dominant players in the enterprise software market, both with multi-billion dollar revenues. The model for both companies was selling software to be installed on the client's hardware on the client's premises (hence "on-premise"), often assisted by specialists from the software provider.

2.2 The personal computer and client/server computing

Also in the late seventies, personal computers started to become accessible and used by individuals outside of a business setting. These were the days of the Commodore PET, Atari, TRS-80, Apple II and various other machines. Very quickly, software for personal computers started to be pirated, and commercial software producers were very unhappy at this. Bill Gates, cofounder of Microsoft, was an early moralizer against software piracy with his famous **Open Letter to Hobbyists** in 1976 (see Figure 4).

On August 12, 1981 IBM would launch its Personal Computer – the IBM PC. This would launch an era of a PC on every desktop and the demise of the "dumb" terminal screen. The PC would also be key to the introduction of the client/server computing model. The premise of this model was to place certain processing tasks, like field formatting and validation, close to the end-user on his PC and place complex business logic and database administration on a shared server, hence the term client/server computing.

This was the dominant architecture in the late eighties and early nineties. However, the client/server architecture proved to be overly complex to manage and optimize. The protocols between the client



-2-

February 3, 1976

An Open Letter to Hobbyists

To me, the most critical thing in the hobby market right now is the lack of good software courses, books and software itself. Without good software and an owner who understands programming, a hobby computer is wasted. Will quality software be written for the hobby market?

Almost a year ago, Paul Allen and myself, expecting the hobby market to expand, hired Monte Davidoff and developed Altair BASIC. Though the initial work took only two months, the three of us have spent most of the last year documenting, improving and adding features to BASIC. Now we have 4K, 8K, EXTENDED, ROM and DISK BASIC. The value of the computer time we have used exceeds $40,000.

The feedback we have gotten from the hundreds of people who say they are using BASIC has all been positive. Two surprising things are apparent, however. 1) Most of these "users" never bought BASIC (less than 10% of all Altair owners have bought BASIC), and 2) The amount of royalties we have received from sales to hobbyists makes the time spent of Altair BASIC worth less than $2 an hour.

Why is this? As the majority of hobbyists must be aware, most of you steal your software. Hardware must be paid for, but software is something to share. Who cares if the people who worked on it get paid?

Is this fair? One thing you don't do by stealing software is get back at MITS for some problem you may have had. MITS doesn't make money selling software. The royalty paid to us, the manual, the tape and the overhead make it a break-even operation. One thing you do do is prevent good software from being written. Who can afford to do professional work for nothing? What hobbyist can put 3-man years into programming, finding all bugs, documenting his product and distribute for free? The fact is, no one besides us has invested a lot of money in hobby software. We have written 6800 BASIC, and are writing 8080 APL and 6800 APL, but there is very little incentive to make this software available to hobbyists. Most directly, the thing you do is theft.

What about the guys who re-sell Altair BASIC, aren't they making money on hobby software? Yes, but those who have been reported to us may lose in the end. They are the ones who give hobbyists a bad name, and should be kicked out of any club meeting they show up at.

I would appreciate letters from any one who wants to pay up, or has a suggestion or comment. Just write me at 1180 Alvarado SE, #114, Albuquerque, New Mexico, 87108. Nothing would please me more than being able to hire ten programmers and deluge the hobby market with good software.

Bill Gates
Bill Gates
General Partner, Micro-Soft

figure 4 - Bill Gates's Open Letter to Hobbyists from the Homebrew Computer Club Newsletter, January 1976 and server components were often very verbose, causing excessive network traffic and contention. As well, performance over a wide-area network was poor, if the application worked at all. Limitations in management tools made IT operations very manual intensive. Regular operations such as installing a security patch were complicated by the fact that there were potentially hundreds or thousands of client and server machines that needed to be patched.

The client/server model came to a slow demise and the "thin client" become the new architectural choice. The notion of "thin client" indicated that very limited processing occurred on the client machine. Indeed, this was limited essentially to rendering the image on the screen, although some scripting could be embedded in the screen display. The standard architecture dictated an Internet browser on the client machine. In these early days, the Microsoft Internet Explorer (released August 1995) and Netscape (released December 1994) waged a fierce battle for market share.

This browser-based approach made possible the delivery of software over the Internet. An early offering was named "Application Service Provider" or ASP. This was essentially a managed service offering where on-premise software was installed and managed by a hosting provider. The more prevalent delivery model that emerged was the Software as a Service model, as we will explain in the next section.

2.3 The end of Software

Fast forward to 1996, Marc Benioff, then a very successful vice-president at Oracle, had a dream to build enterprise software that would be as reliable and as easy to use as buying books on ***amazon.com***. His software would be accessed "on demand" through the Internet and be sold on a subscription basis (5).

Benioff requests a leave of absence from Larry Ellison to go and pursue his dream and eventually leaves Oracle and goes "all in" in the pursuit of success with this new software model. Benioff takes a long look at Sales Force Automation (SFA), a functional area where Tom Siebel), another Oracle alumni, had a resounding success with his ***Siebel*** software package but whose initial versions were client/server.

In late 1998, Benioff lines up three crack developers with deep experience in SFA (including Parker Davis who is still at Salesforce, currently executive Vice-president of technology). In early 1999, Benioff sets up his development team in his San Francisco apartment and starts the creation of his SFA software under the name Salesforce.com.

Initial press and market reaction was extremely positive and Salesforce was poised to take off in true dot-com fashion. The group at Salesforce created an extremely imaginative marketing campaign to break through all the industry noise. They created the "no software" logo that was the center of their corporate persona. The "no software" mascot regularly showed up in front of the Moscone Convention Center in downtown San Francisco during Siebel and Oracle's annual user conferences.

figure 5 – Salesforce's No Software logo

In his book ***Behind the Cloud***, Marc Benioff describes their first thousand user sale to SunGard. What SunGard listed as their requirements applies to many SaaS users (see box).

SunGard's Purchasing Criteria

Security: data couldn't get leaked or lost;
Scalability: it had to grow with the company;
Reliability: it had to be accessible 24/7;
Performance: it had to work right away;
Integration: it had to integrate with the back-office systems;
Customization: it had to look and feel like a Sungard edition.
And finally: "We want to pay-as-you-go business model. It aligns your company's goals with our goals."

Source: Behind the Cloud: The Untold Story of how Salesforce.com went from idea to billion-dollar company-and revolutionized an industry (5).

Salesforce innovated in many areas of sales, marketing, customer support, software release management, etc. as they legitimized the Software as a Service model to the corporate world. And the reward of being true innovators is that they created their own market space in which they become dominant players. Today, Salesforce has added other services in addition to SFA, as well as a PaaS offering: **force.com**. They have also nurtured an eco-system of thousands of partners who make add-on applications to Salesforce that are downloadable from Salesforce's App Exchange.

With 19,000 employees and close to 7 billion dollars in revenue, Salesforce has convincingly shown the pertinence and the viability of the SaaS model.

Other companies followed suit and a growing number of SaaS solutions began to appear on the market, with the early adopters coming from enterprise business applications such as Customer Relation Management (CRM), Enterprise Resource Planning (ERP), human resources and human capital management and payroll. Other software categories followed the SaaS trend, such as:

- IT Infrastructure Applications such as security, monitoring, logging testing, email archiving, source code management, defect tracking and service desk management;
- Data management: Business intelligence, Database as a Service, dashboards, data mining, KPI & Analytics;
- Productivity tools: collaboration tools, development tools, surveys, email campaign tools, content management and social media.

Today, SaaS is incorporated into the strategy of almost all leading software suppliers.

The shift from client/server computing to a server-side model, such as SaaS was driven by ease of management: no hardware purchases, no installation, limited configuration tasks, no backups, ubiquity of access, automatic scaling, easier software deployment and patching.

Even Oracle, one of the largest vendors of on-premise software in the world, encouraged, tested and embraced the Cloud and the SaaS model of delivering software. At the creation of Salesforce, Larry Ellison, Oracle's CEO invested $2 million of seed money and joined the Board of Directors.

In 1998, Larry Ellison invested $125 million in the creation of NetSuite, a SaaS ERP. In 2016, Oracle announced its intention to purchase NetSuite for $9.3 billion.

In 2005, Timothy Chou, then President of Oracle OnDemand echoed Benioff's message and declared the End of Software in a book.

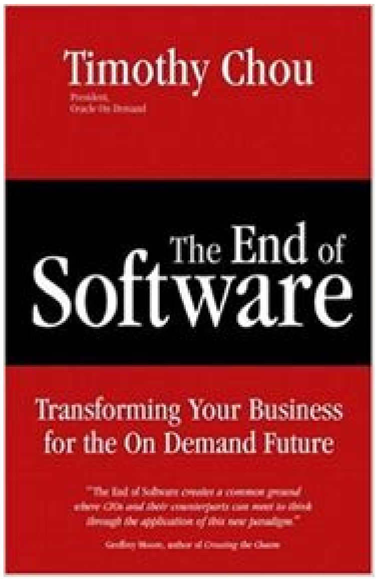

Timothy Chou
President,
Oracle On Demand
The End of
Software
Transforming Your Business
for the On Demand Future
"The End of Software creates a common ground
where CIOs and their counterparts can meet to think
through the application of this new paradigm."
Geoffrey Moore, author of Crossing the Chasm

figure 6 – The End of Software, Timothy Chou

In recent years, Oracle has invested significant money and efforts in creating SaaS versions of many of its products under the marketing label Oracle Cloud.

In this section, we have seen the evolution of software models to arrive at the SaaS software model. In the following chapters, we will characterize the SaaS model and explain in detail the differences with the on-premise model.

ADOBE's brave switch to SaaS

In 2013, Adobe made a 180° turn on their business model. For over 20 years, Adobe's Photoshop was the gold standard for photo editing and image manipulation. However, Adobe was at a crossroads. The company was still profitable, but revenue growth was achieved by increasing prices on the packaged software, not from growth in new users.

So, Adobe announced Creative Cloud, the subscription-based replacement to its Adobe Suite of products (6). So instead of a $700 one-time purchase for Photoshop, customers were offered monthly subscriptions, starting at $10 a month. Initial response was on par with announcing a new formula for Coca-Cola: customers were furious and launched several petitions, even trying to get the White House involved. Twenty percent of Adobe customers said that they planned to abandon the service. Common complaints were from users who said that they did not necessarily want every upgrade automatically while others said that they didn't want the entire suite of apps that was bundled in Creative Cloud.

However, the results over the following quarters proved that Adobe probably saved their company by moving to a more relevant business model (7). At the end of fiscal 2014, Adobe reported $1.7 billion in annualized recurring revenue for Creative Cloud, an increase of 70% over 2013. In Q4 of 2014, about two-thirds of the company's revenue was from recurring sources, up from 44% the year before. Bottom line: Adobe's share price nearly doubled over the two years.

In Summary

In summary, SaaS has the following characteristics:

- some form of **multitenancy**, almost always. We describe the different technical implementations of multitenancy in Chapter 5 when we describe the technical challenges of SaaS;

- **subscribe and use immediately**; as opposed to commercial on-premise software where the installation and configuration can take weeks, or even months;

- **no download** or installation of software; all access is done through a standard web browser or an API;

- **use as designed**; all tenants run on the same code base, therefore the only customizations allowed are the ones that have been designed into the system as configuration settings.

3. ANATOMY OF A SAAS PROVIDER

"There has been something fundamentally wrong with the technology industry for a long time. There's been an unnecessary burden placed on customers to ensure that enterprise software delivers on the vendor's promised results….Maybe if these software applications worked, it would be okay. The problem is, they don't.

Customers don't want to develop [applications] but they do want to make the system completely for them, make their technology fit their business and not the business fit the technology."

Marc Benioff, Chairman and CEO, Salesforce.com

A software house is a software house is a software house…maybe. Does a software house developing an on-premise product function any differently from a software house that is developing a SaaS product? Although we will find many functions with similar names in the two organizations, there are some fundamental differences that are important to understand.

In this section, we will present the major functions of a generic software provider. Then we will present the key differences when the software provider operates under a SaaS model.

Benioff's quote at the beginning of this chapter illustrates the "SaaS promise" or at least the expectations that a SaaS Tenant has vis-à-vis the SaaS service: they expect a software service that actually works, meets their business requirements, is highly usable, through a Web Browser or an API, from anywhere and is always available.

Therefore, the SaaS Provider must be organized to meet this promise.

3.1 Major functions of a software provider

When you open a carton of milk to pour it on your cereal in the morning, you don't give a second thought on the supply chain that allowed that carton of milk to appear in your fridge. The average buyer of software puts in just about the same amount of thought when they break the shrink-wrap on their new copy of Microsoft Office. There are multiple steps before a CD can be pressed or a software package can be made available for download.

Obviously, there is coding going on somewhere, but many more functions are required, both before and after the coding happens. These functions are illustrated in the figure below. We will describe them in more or less sequential order. In a large software company, such as Oracle or Microsoft, each of these functions can represent a Group Vice-President with several hundred-people reporting into him or her. On the other hand, a startup with two founders will still have to cover these functions, with each founder playing multiple roles.

3.1.1 Product Management

All software products start off with an idea or a problem worth solving. Product Management is the custodian of that idea and the keeper of the product roadmap for its development.

The Product Manager is responsible for defining the product, deciding which features and functions should be included in each release. The Product Manager builds the business case for investing in product development; in other words, he will need to convince management that if we spend X dollars building a specific feature, then we should be able to sell Y copies of the software at a price of Z dollars.

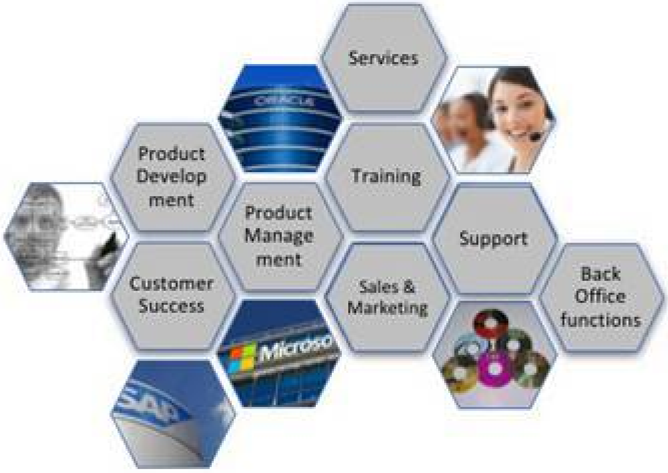


figure 7 – Standard software publisher functions

Since he has a quasi-P&L ("proft and loss") responsibility, the Product Manager will have a strong voice in pricing and packaging decisions. For example, do we publish a low-cost "starter" version with limited functionality, hoping that many users will later upgrade to the "pro" version?

The Product Manager will spend a lot of time listening to customers, gathering feedback on the current product, testing ideas for new features or products, and listening to their suggestions for the evolution of the product.

The Product Manager will also monitor the competition: what distinctive features have they brought out? what are their weaknesses that we need to market as key differences? what are the reasons why we lost out in a specific sales situation? is our pricing competitive?

Internally, the Product Manager acts as an arbitrator of sorts, bridging gaps between Sales & Marketing, the Product Development teams and Management.

As phrased by Bouzid and Rennyson: “Product [Management] determines the right thing to build, and engineering [Product Development] determines the right way to build right.” (8)

3.1.2 Product Development

This is the core function within a software firm. Different development methodologies and ways of organizing this group are beyond the scope of this book. Suffice to say that there will typically be some sort of architecture activity, development itself, maintenance and bug fix, documentation, quality assurance and testing and release management.

3.1.3 Sales & Marketing

It is not sufficient to have a great product, you still must make people aware that your product exists and that it can solve a problem that is worth solving for them. Marketing’s role is to raise awareness and to generate leads.

Depending on the type of software solution and the target market (B2B, B2C, B2G[4]) there are various sales strategies that can be used. For example, an app placed on Apple App Store, Google Play or the Salesforce App Exchange should benefit from a digital marketing strategy. A complex ERP product will require a “solution selling” approach and possibly a lengthy RFP process. In between, there are multiple variations.

The Sales team is usually organized according to two popular models: Inside Sales and Outside Sales. An Inside Sales team will function from inside the walls of the software provider. This is done with an outbound call center, online chat assistant, webinars, email campaigns and follow-up emails. In other words, no frequent flyer points for these guys. If the software requires a “high touch” sales approach, then Inside Sales’ role will be to prioritize leads and coordinate sales meeting for the Outside Sales representatives.

Outside Sales, on the other hand, are based on face to face meetings, presentations, demos and ultimately, closing deals. A costlier model, but in certain situations the only one that works.

3.1.4 Training

Many software providers of business software will have a Training function. The more complex the software, the more importance this group will have. Obviously, providers of simple apps can rely on user institution, a bit of online help and a Youtube video or two to get over the training hurdle.

In addition to being a source of profit, adequate training will often be a key success factor for user adoption and the long-term success of the software installation. Some software providers may offer certification programs as way of promoting excellence in the user or consulting communities. Additionally, training can be "out-sourced" to an eco-system of training partners or resellers.

3.1.5 Services

The Services function consists of consulting staff who are experts in the technical or functional aspects of the software. They will be part of an implementation project designed to assist the customer in installing and configuring the software package. Or they can be called in to solve a performance or upgrade issue later.

Just like the training function, Services groups are more prevalent in complex business software providers. This function can also be covered by one or more implementation partners.

3.1.6 Support

The Support function is there to assist users who encounter any issues using the software. The issues handled can range from password resets to "how do I do…" to reporting bugs. The Support agreement will specify the types of support services provided, the communication means (phone, email, dedicated contact, etc.) and the response time provided.

Time to respond and time to correct are often two different metrics. If the software is supporting a critical business function, then you should special attention to the terms concerning situations where the software is completely unusable; often referred to a Severity 1 issue or a Business Stand.

3.1.7 Customer Success

Customer Success is a role designed to help get the client on board, using the software effectively and keeping them engaged. This function nurtures a long-term relationship with the customer. In other words, their objective is, at least, to have the client renew his support contract upon expiry and, at best, convince the client to increase his commitment through an upsell of new licenses or new features.

The Customer Success function is sometimes assigned to the Sales team, especially in larger accounts where a designated Sales representative is responsible for all activity in a given account. This is often referred to as a “named account”.

3.1.8 Back Office functions

These are the support functions that are common to most organizations, such as human resources, accounting and payables, legal, etc.

3.2 SaaS specific functions[5]

As we mentioned in the Introduction, SaaS has the basic characteristics of the cloud computing model, that is: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. But more specifically, what are the SaaS characteristics that influence the organization of a software company who embraces the SaaS model?

Infrastructure and operations. In the SaaS Model, the supplier is responsible for all infrastructure decisions and is responsible for the operation of that infrastructure. This includes hardware & software procurement, software deployment & configuration, performance monitoring, security patching, scalability, capacity planning, maintenance, backups, disaster recovery planning & testing, etc.

Constant customer interaction. When a client acquires an on-premise software, he may never have any further contact with the software provider, except when it is time to renew his yearly maintenance contract. In the SaaS model, the SaaS provider is required to deliver a service to the customer every single day. Therefore, there is an ongoing, often daily, relationship between the SaaS supplier and his user community.

Meta-data. In the "on-premise" model, once the software is installed, the software provider usually has little or no insight into how the software is being used, or even if it is being used at all! However, since the SaaS supplier controls the code base and the infrastructure, he has the ability to collect meta-data on usage levels and even use of particular features by the client base. If his application is properly instrumented to collect meta-data, the SaaS provider can get immediate feedback on usage of new features as soon as he has deployed them.

Software release planning. In the on-premise model, the Software supplier issues new releases of his software on CD or electronically, but has no control on when his clients will actually upgrade and start using the new version. In the SaaS model, however, the SaaS supplier controls the software upgrade plan. Typically, there is some level of multitenancy, which means that the SaaS supplier will utilize the same code base for all clients. Although not strictly adhered to by some vendors, utilizing a single code base is the only way to achieve scalability and maintain a large customer base.

If the SaaS provider can achieve the objective of publishing software upgrades that are seamless and with minimal impact to the ongoing operations of his installed base, then he will be able to release new versions at a much higher frequency than an on-premise software provider would be able to do. In the early growth stages, the SaaS provider can gain significant benefits from a “release early, release often” strategy. (9) But what holds true in on-premise, still applies to SaaS: “your reputation is only as good as your last release”.

To accommodate these differences, we introduce 4 new functions in our software supplier model (see Figure 8):

- IT Operations;
- DevOps;
- Security & Compliance;
- Metrics.

In addition, we will highlight some changes to the functions that we presented previously in the on-premise model.

3.2.1 Adjustment to existing functions

The whole “tone” of a SaaS organization should be different from that of an on-premise software shop. To reflect that, we highlight three existing functions that need to adapt to the SaaS model.

Support. Since the SaaS provider controls much more (upgrade cycles, infrastructure, configuration choices, etc.), the Support organization must have access to much client context information to do its job.

Where the Support organization would say to an on-premise customer: *“send us a dump so we can reproduce the problem”;* in the SaaS world, the customer cannot send a dump since he does not have control or access to the software or the infrastructure. Indeed, the client's expectation is that the Support group should be aware, pro-actively, that an abnormal situation has occurred.

Product Management. Product Managers should have access to usage metrics that reveal how the client base is actually using the product. This should be valuable input into deciding where and how to improve the product, rather than relying on client interviews and anecdotal evidence.

This reminds me of a situation at a SaaS company where I worked: the Product Management group insisted that our product needed a specific feature, otherwise the competition would "blow us out of the water". We dedicated an entire release to get the requested feature into production. Three months after the roll-out of the new feature, we could confirm that all of 6% of users were using the new feature.

Services. Being a SaaS product, all the "heaving lifting" of configuring, acquiring, installing hardware and then installing software is taken care of *de facto*. Most successful SaaS products are highly configurable, therefore the Services component to get "up and running" should be minimal.

The SaaS Services group will be composed of experts in configuring the software to have it meet the business and process requirements of a specific tenant. In addition, two more "traditional" services rendered by the Services group are integration, for example to link back-end systems, and building custom reports.

The main objective of the Services group should not be to maximize Services revenue (but breaking even is always nice), but getting Tenants configured and operational as soon as possible, with a high degree of reliability and success. Getting Tenants operational as soon as possible is what ultimately contributes to your revenue numbers.[6]

Finance. Billing in the on-premise model is typically a one-time invoice for the licenses and a yearly invoice for the maintenance contract. In SaaS, a different model may apply. In most cases, the SaaS provider will take monthly online payments, through a credit card. Smaller SaaS Providers will connect into a SaaS Service (what else?) such as Stripe to execute their credit card billing. In some B2B situations with large enterprises, the SaaS Tenant may insist on physical invoice with an annual payment.

The SaaS Provider also needs to determine what happens if the Tenant doesn't pay. Does the SaaS Provider immediately suspend the service? Send an email reminder with a 30-day grace period?

3.2.2 IT Operations

Central to successfully delivering a SaaS service is an IT Operations group that is up to the challenge of operating the software product 24/7 with 99.999% availability, all the while meeting the CFO's request to reduce operating costs.

Some of the key roles of this group include:

- Engineering
- Hardware installation
- Software installation and Systems Administration
- Networking
- SaaS Software configuration
- Level 2 and 3 Support
- Release Management

Depending on the size of the organization, each of these functions could represent a separate team in a large company, or it could be a single person doing it all in an early-stage startup.

Another function that is often assigned to this group is Performance Testing & Performance Monitoring, which takes on more importance in the SaaS model, especially when specific contractual SLA obligations exist. In the on-premise model, the response to a performance issue is often to “throw hardware at the problem” by adding another server, adding memory, etc. In the SaaS model, such actions can have even greater ramifications. The cost of additional infrastructure can potentially be multiplied by the number of tenants. The cost of adding a server can potentially double if a Disaster Recovery infrastructure also needs to be upgraded. The capital cost of adding those servers can impact the financial projections of the SaaS Provider and his profitability numbers.

It is this group that needs to make the key decision on determining the best infrastructure on which to run the SaaS service. We describe these technical design choices in Chapter 5 – The technical challenges of SaaS.

We will also go into more detail on the operational aspects of IT Ops in Chapter 6 – SaaS Service Delivery.

3.2.3 DevOps

DevOps is the contraction of “software DEVelopment” and “information technology OPerationS”. It refers to a set of practices that promote collaboration and communication between software developers and IT operations personnel while automating software delivery and infrastructure changes. It aims to implement practices and a culture where building, testing, releasing software can happen more rapidly, frequently and reliably.

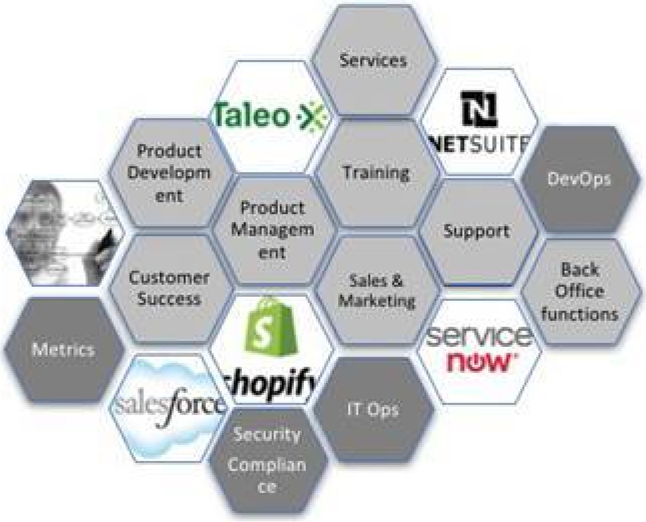


figure 8 – SaaS provider functions

Just like cloud computing, DevOps has a very broad definition. In a typical on-premise software shop, DevOps will often refer to “continuous integration, continuous delivery and agile development.” In the world of SaaS, it takes on a much broader meaning.

Some organizations create a DevOps group and staff it with smart people that understand both systems administration and software development. Unfortunately, this is only half the story since this often just creates another silo. The idea behind DevOps is to break down silos. DevOps is more of a culture shift that reconciles the conflicting nature of work between IT Operations, Software Development and Quality Assurance. IT Operations seek stability while Developers seek change. In between, QA Testers seek risk reduction. These three objectives are difficult to align.

To deliver a stable and reliable product, there needs to be a tight collaboration between Software Development, Quality Assurance and IT Operations. The objective is to garner collaboration between these groups to enable shorter development and release cycles. The DevOps processes should also shorten the feedback cycle between the reality of the product in production and the integration of improvements by Software Development. This is important since the performance and the scalability of the software in production will impact both client satisfaction and profitability.

The DevOps approach for a SaaS provider means that Development Team will consider the ease of deployment and the operability of the software that they are producing. It is no longer a question of throwing software "over the wall" to the guys in Production. Development, Testing and Ops all share an equal responsibility for delivering quality software in Production and eliminating technical debt. This effectively eliminates the finger-pointing and blame game between IT Operations and Development when there is a service interruption.

Having Development staff "on call" and handling service interruptions "shoulder to shoulder" with IT Operations fosters a rapid awakening as the impact their code has in production. This awakening will lead to Development delivering code that is more operable and supportable by the Ops team.

3.2.4 Security & Compliance

The challenge of security and compliance has increased by several orders of magnitude with the widespread adoption of cloud computing. Not long ago, most enterprise software was installed within the walls of a corporation and access from outside was forbidden or strictly controlled. With the advent of cloud computing and the interconnection of systems belonging to clients and suppliers, the security threats can come from multiple sources.

The Security and Compliance group within a SaaS provider plays a key role in ensuring that the SaaS tenants maintain their trust that their data is safe and secure. Some of their key tasks include:

- Ensuring that the architecture of the service ensures data privacy even though there may be shared components;
- Ensuring that only authorized users can access the system, and that authorized users only see the data that belongs to them;

- Hardening of the infrastructure to ensure that there are no exploitable weaknesses in the network or computing devices;
- Running periodic penetration tests to ensure that there are no residual weaknesses. Penetration testing historically targeted network devices (example: open ports) or known vulnerabilities in operating systems (example: unpatched security weaknesses). However, in the SaaS model, equal attention needs to be given to the SaaS software itself which is, by design, exposed to the public Internet. Therefore, vulnerabilities such as SQL injection, cross-site scripting or older/unpatched development frameworks need to be looked at closely;
- Establishing the compliance profile of the organization and obtaining the required certifications.

3.2.5 Metrics

This function is often not a dedicated group, but a shared responsibility among many stakeholders within the SaaS provider's organization.

Information gathering and information sharing are key activities within a SaaS provider. We will illustrate this with two examples:

- An IT Ops technician receives an outage alert for a client, quickly followed by an escalation request for this same issue. Before initiating the escalation, the technician would like to validate:
    - How long has this client been operating in Production? What version is he running and when was his application instance last upgraded?
    - Has this client logged previous tickets for this type of outage? Is the client on a "special care" list in Support?
    - Are all users from this client prevented from accessing the system? only a category of users? only users from a specific geography?
    - Is this an important client in terms of revenue? Is he behind in his payments and we were just about to shut down his service?

- A Sales representative is about to make his regular six month call to an existing client who will be renewing soon. Before going to the meeting, he would like to know:
    - What is the usage at that client? (number of users, number of logins, number of transactions?) Is usage increasing or decreasing over time?
    - Has the client had any performance issues? What is his average response time? any outages?
    - Has the client logged any support tickets?
    - Has the client logged any feature requests?
    - How much revenue have we acquired in the past year?
    - Is the client late with any payments?

Collecting this information is time consuming, as different groups (IT Ops, Support, Finance, Product management) all have their own systems and information sharing is done by building Excel spreadsheets. Overcoming these silos of information can be a challenge within many SaaS organizations.

I was once called in to do a strategic review of SaaS operations at a Silicon Valley software company. In preparation for the kick-off meeting, we learned that a member of the staff had spent three weeks full-time collecting data and populating Excel spreadsheets, most of which were out of date on the day of meeting. It is surprising to see how many companies are still running their business on Excel (see figure below).

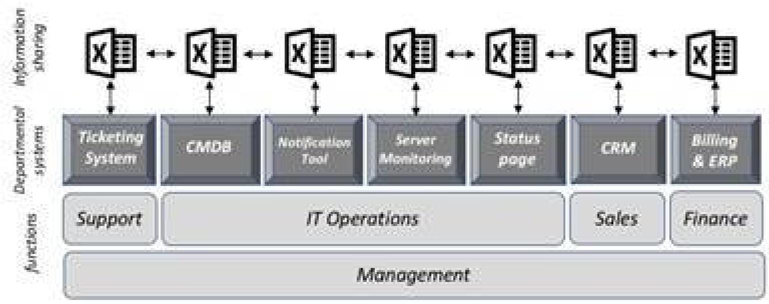


figure 9 – Running a SaaS business with Excel

Having a solid strategy for collecting and managing metrics allows the SaaS Provider to be much more confident in publishing key SLA metrics to his SaaS Tenants. Transparency will result in increased SaaS Tenant confidence in the service.

Having a single source of truth for managing key information also fosters communication and collaboration between the different groups within the SaaS Provider's organization. There is a Danish proverb that goes something like this: *"When you have a clock in your house, you know the time. Once you get two clocks you are no longer certain."* The same can be said of Excel spreadsheets.

3.3 Hybrid models

We sometimes encounter a hybrid model in software companies that were previously publishing an on-premise product and have decided to transition to a SaaS model. In order not to alienate their existing on-premise customer base, they will continue to support the on-premise product while selling their SaaS solution in parallel.

This situation is far from ideal. It can cause confusion and uncertainty among the client base and may even motivate existing client to start shopping for an alternative product. The confusion can be acerbated if there are two separate sales teams for on-premise and SaaS or if the Sales compensation plans skew the sales pitch one way or another.

The hybrid solution can also generate conflict internally since the Product Development group will eventually need to maintain two code bases and some feature requests that are specific for the SaaS product, such as instrumenting the application, will not generate any benefits in the on-premise product. In most hybrid situations, the on-premise installed base outnumbers the growing SaaS business, such that the on-premise product may get the most attention (and development dollars).

As for the Support group, one must have two support scripts for on-premise and SaaS clients. The SaaS Provider must avoid, at all costs, having the Support technician ask a SaaS customer to send in a dump or error log so that Support can replicate the problem.

4. THE BUSINESS PROBLEM OF SAAS

"Concentrate on providing value. The economics will take care of themselves as long as you have the courage to price your product at what it's worth."
- Peter Gassner, Founder & CEO Veeva.

In theory, a SaaS business produces a steady revenue stream, removing a lot of the uncertainty of an on premise licensed software business model. Despite this, many SaaS businesses fail. Why? The secret lies in the numbers aspect of your business, that is why you need to take the time to calculate some critical metrics and interpret them correctly.

In this section, we will present the basic economics surrounding a subscription based revenue model. We introduce the key financial indicators and their relationship to building a successful SaaS business. Finally, a discussion of forward indicators of change and their importance.

Almost all cloud companies get started by an entrepreneur who feels he has a great idea. In the gestation and startup period of the company, a lot rests upon the “gut feeling” and instincts of the entrepreneur. This belief in one’s self, and in the project, is important. It is often something that makes an enormous difference when pitching to a VC, negotiating a line of credit, hiring a strategic resource, etc. However, numbers are also important in understanding the strengths and weaknesses of a specific business model. Therefore, mastering the “business problem of SaaS” means mastering the numbers that define a subscription business model.

4.1 The on-premise software business model

The standard on-premise licensing model produces a lump sum payment upon purchase, along with possible maintenance fees usually paid on an annual basis.

Therefore, software companies using this licensing model will build a sales forecast and assign quarterly sales quotas to their sales teams. Predicting a steady revenue stream can be challenging: the results achieved in one quarter have no influence on the results that might be obtained in the following quarter. At the start of each quarter, license revenue starts at zero and the sales team needs to close new deals to meet the forecasted sales number. The sales team needs to maintain a steady pipeline of qualified leads to close new deals every quarter.

It is the quality of this pipeline that will determine the future revenue streams.

An additional challenge for the CFO of an on-premise software company is that revenues are not equal quarter to quarter. Very often the revenue numbers in the last quarter of the fiscal year are higher than the other quarters as sales representative accelerate deal closures to meet their annual sales quota and receive their plane tickets to the "President's Club" retreat for top sales representatives. Notoriously, clients will often delay their purchases until the 4th Quarter since it is in the last month before year end that the greatest discounts are often given.

For most enterprise software offerings, a maintenance contract is mandatory. Typically, the maintenance fee will be in the range of 18 to 22 %. Over time, with a sufficiently large customer base, this predictable and relatively stable (other than non-renewals) can add some predictability to an on-premise software company's results.

For example, in Oracle's quarterly results ending on November 30, 2016, upgrades & maintenance revenues represented $4,777M on total revenues of $9,035M. However, this probably represents an exception in the industry. Oracle being in business for a long number of years in addition to being a leader in their market segment during that time. For a software company in the process of building its installed base, maintenance revenues will grow slowly over time and will remain insignificant, in the early years, with respect to one-time license revenues.

4.2 The annuity business model

The lack of predictability of the license revenue stream for an on-premise software company explains, in part, why CFOs and financial institutions alike love the annuity business model. It offers predictable revenue streams.

When I was working at a major HCM SaaS vendor, 85% of the revenues expected for the upcoming quarter was already "guaranteed" by ongoing contracts. This made financial planning all the easier. By limiting up and down swings in revenue, it was a lot easier to plan for hiring, equipment purchases and other capital expenditures. In an on-premise model, there is more risk of last-minute replanning of expenditures because the sales results were not according to plan.

The revenue stream for a SaaS business is what is called in financial terms an annuity model. Instead of receiving the total value of the contract up-front in one lump sum, the value is spread out in a series of monthly or yearly payments. The most common example of an annuity is a pension. When a person retires, they don't receive the lump sum of the value of their pension fund, but will receive a regular monthly payment whose value is based on several factors such as the pension fund amount, age of retirement, life expectancy, interest rates, etc.

Types of recurring revenue.

John Warrilow) in his book Built to Sell (10) describes several different business models for a recurring revenue business. Each model is presented in order of increasing value:

6. Consumables: Warrilow gives the example of toothpaste. He always buys the same brand, therefore repeatable business for the manufacturer, but it is very easy for hom to switch to another product or be swayed by an apparently more attractive offer.

5. Sunk Money consumables: As an example, if you buy an HP printer, you will become a regular customer of HP printer cartridges. Switching to a competing product requires more effort since you need to purchase a new printer, unpack and install it. Witness Nexpresso and Keurig who are exploiting this business model in the coffee business.

4. Renewable subscriptions. The classic example is magazine subscriptions. Other examples are your membership in AAA or the local gym. In these examples, you need to make the conscience decision to renew, usually once a year (as compared to auto-renew subscriptions listed as #2).

3. Sunk Money Renewable Subscriptions. If the customer needs to invest money to access your service, there is an additional barrier to switching to another service. Warrilow gives the example of the Bloomberg Terminal required to access Bloomberg’s financial information feed. Another example is e-readers. If a consumer has spent a hundred dollars buying a Kindle reader, he is more likely to spend his money on buying Kindle books. The same applies to the consumer who purchased a Kobo reader instead.

2. Auto renewal subscriptions. This is model utilized by many SaaS services. For example, after your free trial period, you enter your credit card number and the subscription gets charged automatically every month until you cancel. Examples could be your subscription to a service like GitHub, Dropbox, Pandora, Netflix or Spotify.

1. Contracts. The only thing more valuable than an auto-renewal subscription is a formal contract for a fixed period (in the SaaS world, generally from 1 to 3 years). The cellular phone companies have practiced this form of subscription lock-in with great success.

By targeting a recurring business model that has increased value, you will also increase the value, and valuation, of your SaaS business.

While the predictability of the revenue stream is an advantage in business and financial planning, there is a down side. You have invested a certain amount of sales, marketing and setup costs to bring the customer online. These costs are not reimbursed immediately, but rather paid back over a period of several months. This payback period will depend on how much you spent to acquire the client and how much margin is generated by each monthly payment.

Therefore, each new client that you sign up will eat into your cash, until they stick around long enough to “pay back” the acquisition costs. Your cash flow situation will look something like the next figure.

The depth of this cash flow trough, and how long it will take to reach the breakeven point depends on several factors, such as: the customer acquisition cost, the rate of new bookings, how much margin is contained in the monthly pricing, how long do clients continue to use the service, etc. Counterintuitively, having more success at sales and signing more customers initially doesn’t help. In fact, it makes things worse (from a cash flow perspective) by pushing the cash flow trough even deeper.

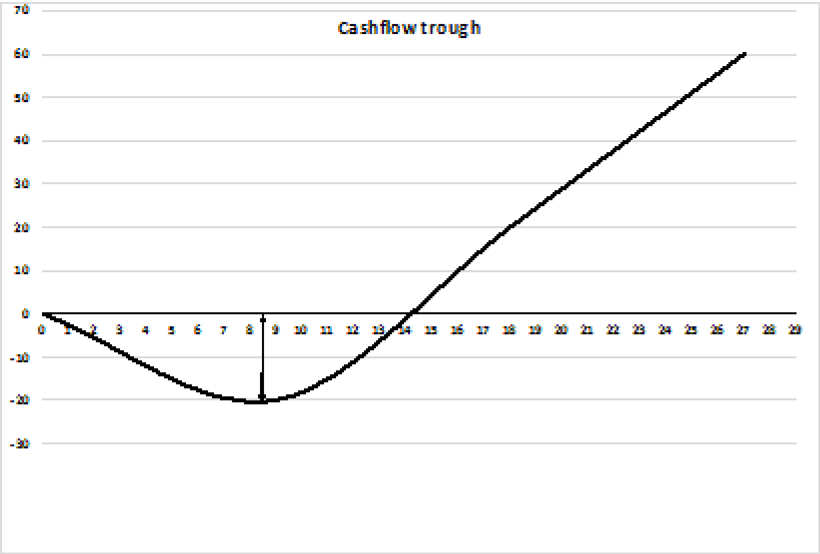


figure 10 – The cash flow trough

In the above example, if it takes almost 14 months to recover the cost of the upfront investments, such that customer only becomes profitable on Month 15. If the client cancels the service on Month 12, or doesn't renew the service for another year, then that client will create a deficit for the SaaS provider. In a such a situation, the SaaS provider may never get out of the cash flow trough and will start a *"slow march to death",* that is, moving slowly but surely to the point where they run out of cash.

There are several strategies for not getting caught in the cash flow trough. First is to work on controlling expenses and trying to conserve cash: slash expenses, negotiate better payment terms from your suppliers. The second is to be disciplined on the revenue side: ensure your pricing is right, track receivables closely, minimize "write-offs" for non-payment, look at financing mechanisms that will recognize your future cash-flows. Third, leverage cloud services (SaaS, IaaS or PaaS) that scale with your business wherever possible and re-examine each of these choices as your business reaches scale.

Getting your pricing right can be difficult, especially when you have just launched your product. If you sign a new client for a one year contract with a renewal for 2 more years, then a bad pricing decision will haunt you for next 36 months.

In order to avoid bad pricing decisions, you should calculate the target profitability over the life of the contract, including all costs; review the competition's pricing; determine the break-even point for every deal; how many months are needed to reach payback of the acquisition costs?; establish different service bundles to meet all price points (example: bronze, silver, gold).

This discussion is based on a monthly revenue model, which is more or less, the standard in the industry. Either the SaaS Tenant pays month to month and can opt-out any time, or they sign a one year contract but still pay month to month. But what if they pay the full year in advance? Bingo! the cash flow trough disappears and the SaaS Provider is cash flow positive from Day 1 of the sale.

Moving to annual sales model does create more challenges in the sales cycle; the SaaS Tenant will really need to feel committed to the service to pay for one year up front. To overcome this resistance, SaaS Providers may choose to only offer a yearly subscription with an up-front payment. With no other pricing alternative, there can be little discussion. Others may offer a discount for a yearly payment; typically, this will be somewhere in the range of 10% or perhaps one month out of twelve free.

To better discuss the economics of the annuity business model, as it applies to SaaS, we will define several key metrics in the next section.

4.3 Key SaaS business metrics

In this section, we will define 8 key metrics that can be used to analyze a SaaS business. In the following section, we explain the relationship among these metrics and offer some “rules of thumb” that can be used.

4.3.1 Monthly Recurring Revenue (MRR)

MRR is simply the sum of monthly subscription revenue. If you charge for your service by the month, simply add all monthly revenues. If you pre-invoice for a set period, then divide the contract amount by the number of months of the subscription. For example, divide by twelve for a yearly subscription.

Obviously, you need to subtract any discounts that you have given, and convert any contracts in a foreign currency back to your primary currency. You should not add in any one-time fees, since these are not recurring charges.

Why is MRR an important metric? MRR is probably one of the most scrutinized indicators of a subscription-based business model. How much cash comes in at the end of the month allows you to pay your hosting provider, make your payroll, hire a new sales representative, negotiate a line of credit, etc. Cash is always king; especially when you are just starting out.

MRR can also be a key indicator when you are looking for financing since many VCs will value your company as a multiple of your MRR.

To do a more detailed analysis of MRR growth or reduction, you might want to break up the number into different components:

- **New Business MRR**
  MRR of new customers that appear on the revenue sheet for the first time;

- **Expansion MRR**
  An increase of MRR of an existing customer; e.g. upsell. Adding a new service, increasing usage counts, end of a discount period;
- **Contraction MRR**
  A decrease of MRR from existing customers; reduction in usage counts; downgrade to a lower plan;
- **Churn MRR**
  MRR of customers who cancel their service completely or fail to renew;
- **Reactivation MRR**
  MRR of a previously churned customer who returns to the service.

4.3.2 Annualized Run Rate (ARR)

Quite simply, MRR x 12.

4.3.3 Customer Acquisition Cost (CAC)

Customer Acquisition Cost (CAC) is the dollar value that you have invested to get a SaaS Tenant to sign up for your service. In your calculation, you should include all costs incurred to acquire customers, not just the direct costs. Therefore, this metric should include personnel costs (inside and outside sales organizations, marketing manager, account executives, etc.), advertising (social media, Google AdWords, ad space on blogs and other websites, mailing lists, etc.), and marketing costs (content marketing, website design, lead scoring, etc.)

$$\boxed{CAC = \frac{Sum\ of\ all\ Sales\ \&\ Marketing\ expenses}{number\ of\ new\ clients\ signed}}$$

CAC is an important metric since it will allow you to evaluate whether your pricing is adequate and how long it will take to take recover your CAC investment. In other words, it allows you to estimate the depth of your cash flow trough and your financing requirements.

4.3.4 Average Revenue per Account (ARPA)

Sometimes referred to as ARPU (average revenue per user or per unit) or ARPC (average revenue per client). Simply calculated as the average MRR across all your customers:

$$\boxed{ARPA = \frac{sum\ of\ all\ customer's\ MRR}{number\ of\ customers}}$$

If a customer has multiple accounts, those should be grouped together and considered as a single account for this calculation.

4.3.5 Churn Rate

Churn is the rate at which customers are leaving your service. You can measure churn in number of customers or in terms of revenue.

$$\boxed{\begin{gathered}Customer\ Churn\ rate = \\ \frac{Nb\ of\ customers\ who\ churned\ in\ the\ period}{Total\ nb\ of\ customers\ at\ the\ beginning\ of\ the\ period}\end{gathered}}$$

Revenue Churn can be calculated as **Gross Revenue Churn** by measuring the losses in revenue, for a given period, divided by the total revenue at the beginning of the period. Revenue churn can also be calculated as **Net Revenue Churn**, where losses are offset by account expansions. For the following example, we will set the period to be a month, thus:

$$\boxed{\begin{gathered}Net\ MRR\ Churn\ rate = \\ \frac{\begin{gathered}Sum\ of\ Churn\ and\ Contraction\ MRR - \\ Sum\ of\ Expansion\ and\ Reactivation\ MRR\end{gathered}}{MRR\ at\ the\ beginning\ of\ the\ period}\end{gathered}}$$

4.3.6 Customer Lifetime Value (LTV)

The Customer Lifetime Value (LTV) is an estimate of the total value (i.e. gross revenue generated) of a client from sign-up to churn. The total revenue that a client will bring to the SaaS Provider will depend on several factors, most importantly:

1) **The pricing**; the SaaS Provider will want to ensure that his tenants are paying the right amount for the value that they are deriving from the service;

2) **How long they use the service**; how long a client continues to use the service, and pays for it, directly influences the customer's lifetime value (LTV).

For example, if a Tenant signs up for a monthly Netflix service at $10 per month, and cancels after 4 months, then his LTV will be:

$$LTV = Month\ 1 + Month\ 2 + Month\ 3 + Month\ 4$$
$$LTV = \$10 + \$10 + \$10 + \$10 = \$40$$

The more financially astute readers will recognize LTV as a discounted cash flow calculation. However, let us present a simpler way of estimating LTV:

$$\boxed{LTV = \frac{ARPA}{Customer\ Churn\ Rate}}$$

Our previous definition of Average Revenue per Account (ARPA) was based on a monthly period. What is important in this basic LTV calculation is that both ARPA and the Customer Churn Rate are from both the same time period for which you wish to calculate LTV.

As with many simplifications, there are several limitations to this basic number. Even though it is only an estimate, it is a good place to start. Be aware that it excludes one-time fees which may be a significant part of your revenue stream. This formula will be less accurate with a small number of tenants. It also assumes a linear churn rate, which may or may not be applicable to your business model and does not account for revenue expansion or contraction during the life of the contract.

A more complete formula for LTV is:

$$LTV = \frac{ARPA}{Customer\ Churn\ Rate} \times .75 + \left(\frac{m\ (1 - Customer\ Churn\ Rate)}{Customer\ Churn\ Rate^{2}} \right)$$

The discount factor of .75 produces a more conservative estimate of LTV to compensate for different patterns of churn. Some SaaS services may have many cancellations after the first month, followed by a smaller churn rate. Other services may have a more staggered churn, aligned with the contract renewal period, whereas some SaaS services truly have a linear churn pattern.

The second term, as proposed by David Skok, allows to include linear account expansion based on a variable ***m***, which is the monthly growth in ARPA per account.

Customer LTV is an important metric because it is a good indicator of product health and customer retention. If your LTV is increasing over time, it is a good indication that your tenants are happy and that they are using more of the service or sticking around longer before churning. However, if LTV is decreasing then the SaaS Provider is deriving less and less revenue from each client and needs to identify the issue.

4.3.7 Average Sale Price (ASP)

Represents the average MRR of a new customer when they first signup:

$$ASP = \frac{Sum\ of\ all\ new\ business\ MRR\ in\ the\ period}{Number\ of\ new\ customers\ in\ the\ period}$$

This is a good indicator of the perceived value of your service by the marketplace and the effectiveness of your sales strategy to drive higher deal size.

4.3.8 Renewal Rate

Just like Churn, we can measure renewal rate in number of customers or in revenue.

Customer renewal rate, or retention is measured as:

$$Customer\ Renewal\ Rate = \frac{Number\ of\ customer\ who\ renewed\ in\ the\ period}{Number\ of\ customers\ up\ for\ renewal}$$

The MRR renewal rate is calculated similarly:

$$MRR\ Renewal\ Rate = \frac{MRR\ of\ renewed\ contracts}{Total\ MRR\ of\ contacts\ up\ for\ renewal}$$

4.4 Why SaaS companies fail

Now that we have defined these key metrics, we can now use these metrics to describe and analyze several situations where the viability of a SaaS company may be compromised.

4.4.1 Leaky buckets; why churn is important

The sales organization in on-premise software house is essentially focused on new customer acquisition (and to a more limited extent, upsells in existing customers). Bringing that culture into a SaaS organization can be a recipe for failure.

If we look at the top half of the following graph, we can see bookings progressing at the steady rate of $50,000 per month. If we were to look only at this number during a management meeting, we would have the impression that everything is progressing well.

However, if at the same time, the company is losing 5% of its customers who leave the SaaS service, there will come a point where the new sales will not be able to compensate for the customers that are churning. At that point, the SaaS Provider will start to bleed money. At 10 % churn, the effect will be even more dramatic.

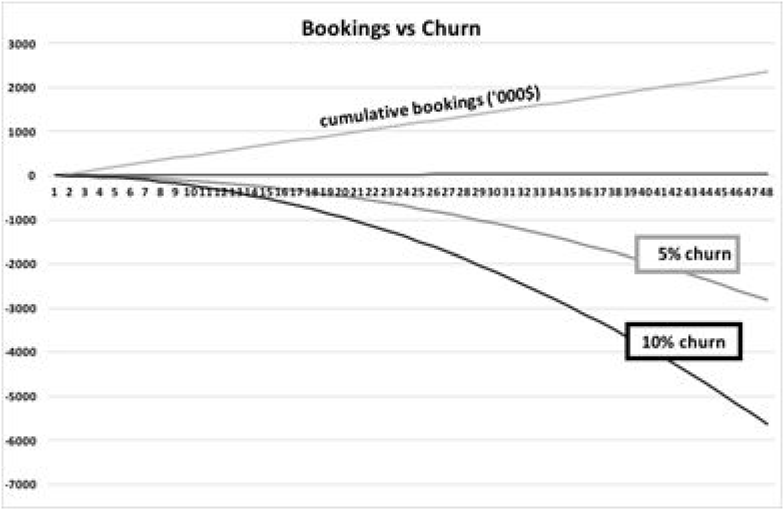


figure 11 – The impact of churn

Our example may be over-simplified, but serves to illustrate two important points:

- Churn, even at low percentage points can have a devastating effect on your SaaS business;
- Reducing churn, even by a single percentage point will have a compounding effect on your results.

The impact of churn: the Salesforce story.

An illustration of the problem that is graphed out in figure 11 is the situation in which Salesforce found itself in 2005. About one year after their successful IPO, Salesforce was reporting 88% bookings growth. They were on a roll and the markets were excited.

However, beneath this apparent success was a looming disaster. The churn rate at the time was **8% per month**. That means that nearly two-thirds of Salesforce's customers were leaving every year.

Salesforce was the first subscription business to reach the scale that it had at the time: 20,000 customers and revenues of $176 million. What Salesforce realized and learned was that they were too focused on acquiring new customers and weren't spending enough attention on retaining the customers that they already had. They couldn't add enough new business in the top of the funnel to sustain real success while they were leaking existing customers out at the bottom at a higher rate. Therefore, the real success of a subscription business, such as Salesforce, is in growing the overall value of the installed base. (11)

Another way to look at churn is to use the leaky buckets analogy. (12) The ultimate goal is obviously to fill up the last bucket with as much water as possible, i.e. long term, engaged, customers. In every SaaS business, there should be a way to qualify an engaged customer. For example, if you are running an online training program, then a fully engaged customer might be someone who has registered for a training program and who has completed the five first workouts. Once they have reached that stage, the probability that they will churn is very low or nil.

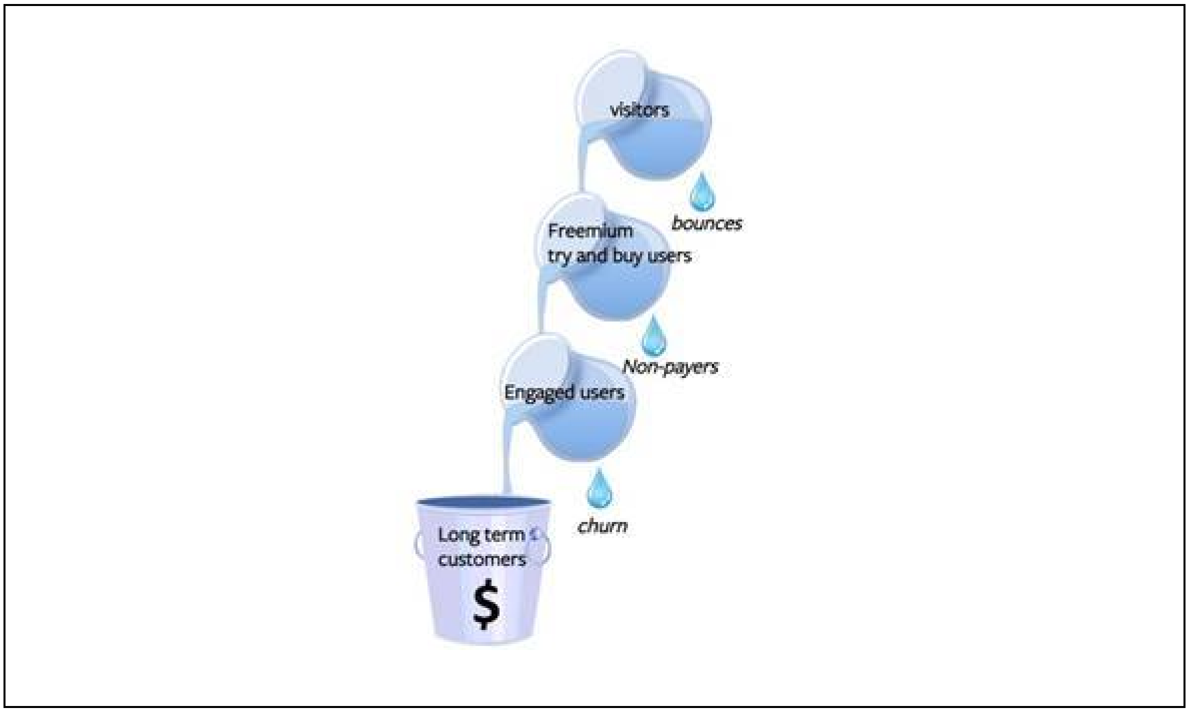


figure 12 – Leaky Buckets

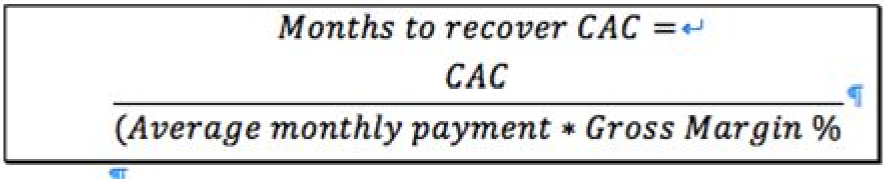
The leaky bucket example also forces us to examine what is happening "upstream". The visitors to your website and the users of your "freemium" or trial version constitute your sales pipeline. Therefore, the amount of water in each bucket is important; it constitutes your future sales potential.

And obviously, we will want to reduce leakage in those buckets as well. For the first bucket, improving stickiness at the public web site and optimization of the hit ratio on the "try now" or "sign up" button. For the second bucket, there are several strategies to keep trial users engaged and motivated to sign up as full users: select notifications, usage suggestions, email updates, etc. For engaged users, we will want to increase usage levels, increase penetration levels within organizations with upselling and premium usage features, etc.

4.4.2 Customer Acquisition Cost

As we illustrated with the **cash flow trough** in section 4.2, one simple reason that many SaaS startups fail is that they pay too much to acquire their clients. A common temptation is to focus on customer growth exclusively without thorough consideration for the CAC.

One measure of sensitivity that is interesting to calculate is the number of months required to recover your CAC:



$$\text{Months to recover CAC} = \frac{CAC}{(\text{Average monthly payment} * \text{Gross Margin } \%}$$

Obviously, the successful SaaS Provider will want to make this payback period as short as possible. Although it will vary from one service to another, a good rule of thumb is to use 12 months as a benchmark for CAC recovery (13).

To further evaluate the sustainability of your SaaS business, you need to measure your LTV and CAC, and then calculate the following ratio:

LTV / CAC.

If the ratio is greater than 1, then you are making money. If it is less than 1, then you are definitely on your ***slow march to death***. Industry analysis suggest the optimal LTV/ CAC ratio should be greater than 3 to compensate the fact that the CAC is spent upfront while LTV is collected as an annuity. The exceptional SaaS companies can have a LTV / CAC ratio as high as 7 or 8. (14) (15) (16)

Two elements of managing the LTV / CAC ratio are:

- Getting your pricing right; this is often the single most element in optimizing LTV;
- Controlling your CAC as you scale; with growth will come added spend in sales & marketing. You need to ensure that added spend is not dragging down your LTV /CAC ratio.

4.4.3 Analyzing churn

Understanding churn, or its opposite – retention, can help analyze the viability of your SaaS business. First, it is important to understand that User churn and MRR churn are two different metrics and tell different things about your business.

User churn provides an indication of your product/market fit, specifically it provides feedback on your company's product, marketing, customer service, and pricing. On the other hand, MRR churn provides feedback on the SaaS Provider’s financial sustainability.

The SaaS Quick Ratio provides a simple metric to measure the growth efficiency of a SaaS Provider and the impact of churn. This metric was devised by Mamoon Hamid, co-founder of the venture capital firm Social Capital.

$$\boxed{\begin{gathered} \mathit{SaaS\ Quick\ Ratio} = \\ \frac{\mathit{New\ MRR} + \mathit{Expansion\ MRR}}{\mathit{Contraction\ MRR} + \mathit{Churned\ MRR}} \end{gathered}}$$

Net New MRR is composed of the four values used to calculate the Quick Ratio. Graphically, it would look like figure 13 on the next page.

In this example:
Net New MRR = \$10,000 + \$24,500 - \$4000 - \$3000 = \$27,500.

We can then calculate the Quick Ratio:

$$\boxed{\begin{gathered} \mathit{SaaS\ Quick\ Ratio} = \\ \frac{\$24{,}500 + \$10{,}000}{\$4000 + \$3000} = 4.9 \end{gathered}}$$

The higher the number for the Quick Ratio, the better. There is some debate on the “ideal” Quick Ratio, but there is a consensus that an early stage SaaS Provider will display healthy growth with a Quick Ratio of 4. (17) (18) (19) However, this is not a hard and fast rule. SaaS Providers serving Enterprise customers will probably experience lower churn rates than SaaS Providers serving smaller clients. And certainly, the Quick Ratio will change as the SaaS Provider reaches scale and maturity.

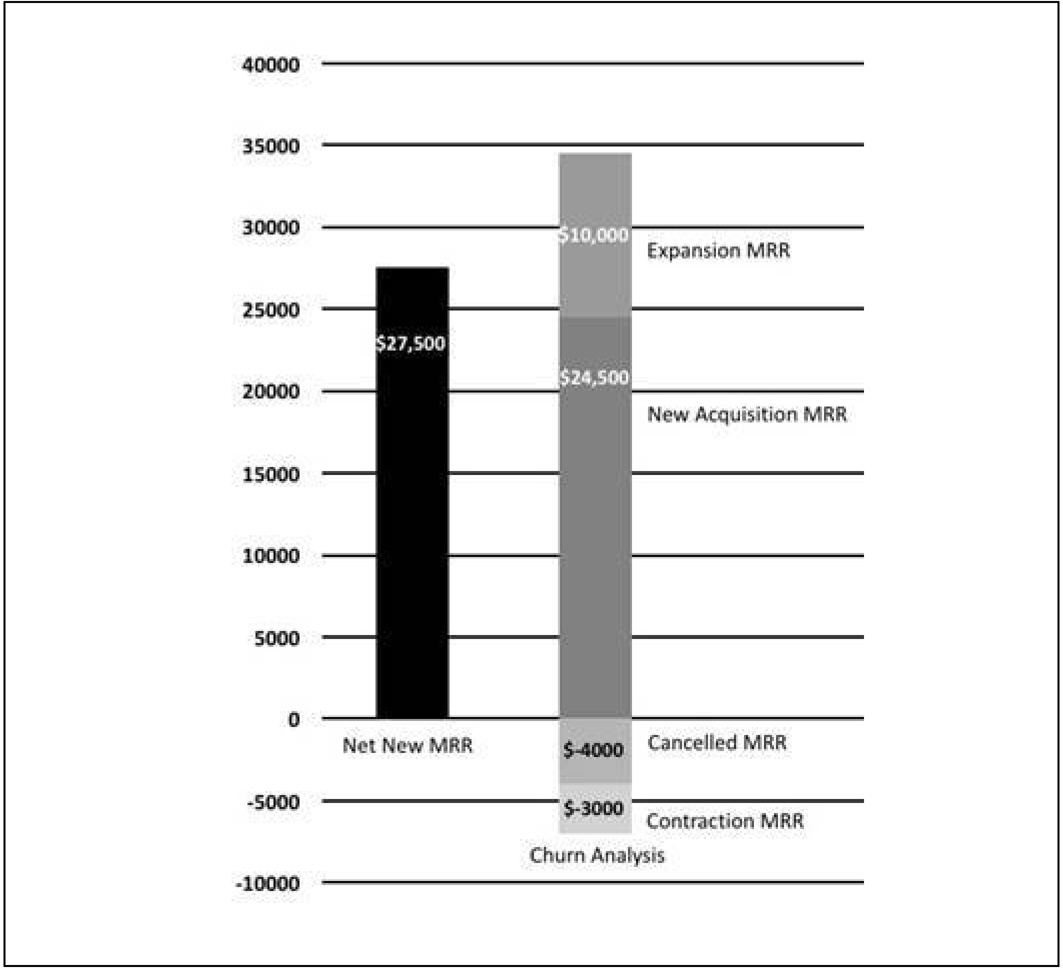


figure 13 – Net New MR Calculation

In figure 13, we have presented a single month. To really understand your churn numbers, you need to graph these over time and observe any changes to your Quick Ratio.

Other factors to consider when analyzing your churn numbers:

- The average retention across a customer's lifetime is not a constant. Look at retention rates for cohorts of newer customers, middle stage customers and long term customers. Retention strategies will differ if you are dealing with an early-stage, mid-stage or late stage SaaS Tenant;

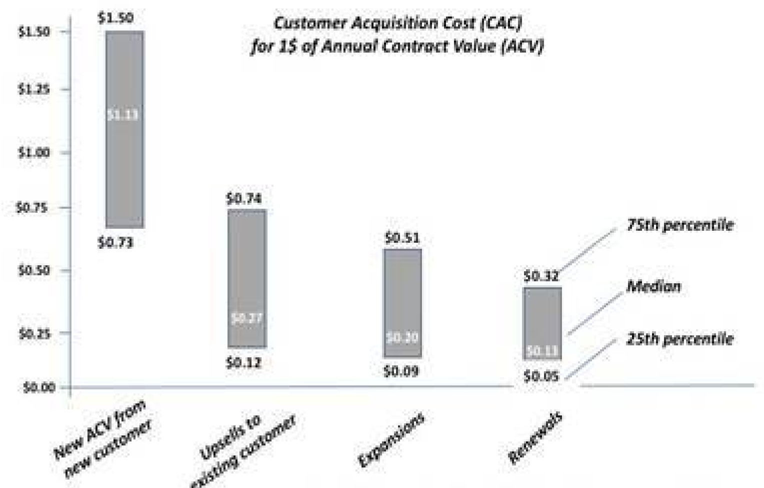
- SaaS Tenants on different pricing plans may perceive value differently. If there is higher churn from customers on a specific plan, then a second look at your pricing strategy may be in order.

4.4.4 Upselling

As we illustrated with the cash flow trough in Section 4.2, your CAC can have a serious impact on the viability of your business. One simple method to reduce the impact of the cash flow trough is not to focus on new customer acquisitions, but to aggressively pursue upsell opportunities with existing clients.

The book Marketing Metrics (20) states that the probability of a successful sale with a new prospect is 5-20% while the probability of a successful sale with an existing customer is 60-70%. Your CAC will also be lower in the case of an upsell, as confirmed by the Pacific Crest Securities 2016 annual SaaS survey (21) (See figure 14).

The median respondent to the Pacific Crest survey gets 15% of new ACV from upsells and expansions. Larger companies rely more heavily on upsells than smaller companies. For example: companies with $40MM to $75MM GAAP Revenue generate 28% of new ACV from upsells, almost three times as much as companies with less than $2.5MM in GAAP revenues. See figure 15.



Source: Pacific Crest Securities, 2016 Private SaaS Company Survey Results

figure 14 – CAC – New ACV versus Upsells

You need to retain and upsell existing customers to have the most profitable model for your SaaS company. In the 2015 Pacific Crest SaaS Study (22), the percent of new ACV from upsells was separated into two groups: the Top 50% of revenue growth and the Bottom 50% of revenue growth. The Top 50% were clearly driving more upsell growth than the Bottom 50%.

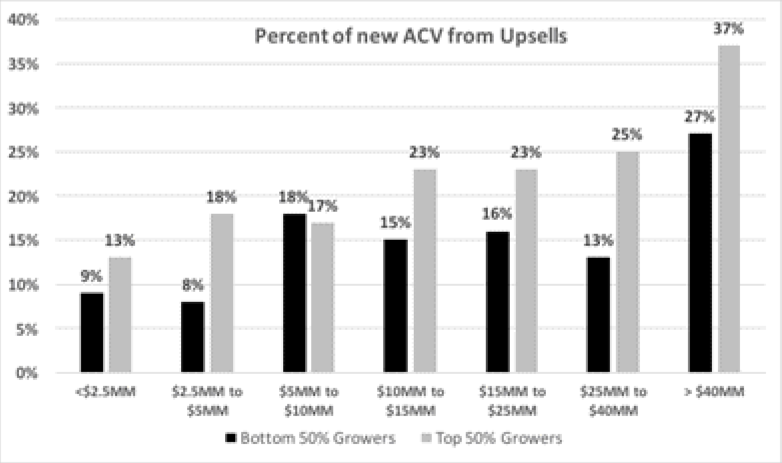


figure 15 – Upsells and Growth

4.5 Forward indicators of change

Economists typically group macroeconomic statistics under one of three headings: leading, lagging or coincident. We can apply the same logic to the metrics used to measure the success of a SaaS business.

Figuratively speaking, looking at leading, lagging or coincident indicators is like looking through the windshield, looking at the rearview mirror, or looking through the side window.

In a typical management meeting, at a typical SaaS provider, you would expect a review of the previous month's results. So, for example: the summary of new bookings for the month and the MRR for the month would be top-of-mind. The issue with indicators such as new bookings, churn, expansion and contraction MRR is that they measure what happened in the past; they are **trailing indicators of change**. Like looking in the rearview mirror, they are not good predictors of how the business will evolve the following month.

A more productive discussion would involve forward indicators of change. Three important forward indicators are:

- Usage;
- Upsells (which we discussed in Section 4.4.3);
- North Star metric.

Usage.

Usage statistics are usually a good indicator of end-user adoption and the stickiness of the SaaS service. Higher usage levels will indicate that churn is less likely in the future and that upsells are more probable.

Usage can be measured in absolute numbers, for example: number of logins, number of messages sent, number of projects created, number of candidates hired, etc. Usage can also be measured by a ratio, such as DAU over MAU. (23)

DAU over MAU is the ratio of daily active users to monthly active users. This metric will indicate if users are only using the service sporadically or if they are truly engaged.

North Star Metric.

Mamoon Hamid, from Social Capital, has coined the phrase “North Star Metric” to describe a metric that measures the value that a SaaS Provider delivers to its tenants. (24)

DAU over MAU is one indicator that could be considered as a North Star Metric. However, an indicator that confirms product/market fit is preferable. For example, for an applicant tracking system, the number of hires made is a good indicator of the value being delivered. For a stock market tracking system, a user that enters 8 different stocks in his profile and checks his portfolio daily over a period of a week is engaged.

A North Star Metric will be application specific and the threshold that confirms that a SaaS Tenant is engaged will be determined over time by observing user retention and usage profiles.

Some other examples of “engaged user” that can be used to calculate a North Star metric:

- Slack: any team that has exchanged 2,000 messages;
- Facebook: a user reaching 7 friends within 10 days of signing up;
- Dropbox: a user that places one file in a Dropbox folder on one device.

4.6 In summary: Six ways to fail

In summary, success can be elusive. It can also be magical, if you find a product/market fit and can ride the wave of accelerated user adoption.

However, there are many ways to fail. Let us summarize six ways to fail using the metrics from the above discussion:

1. If churn exceeds new bookings, **you will fail**.
2. If your customer acquisition costs (CAC) are too high, **you will fail**.
3. If your CAC payback period exceeds your average contract life, **you will fail**.
4. If your tenants do not have confidence in your service, **you will fail**.
5. If you run your business by looking in the rear-view mirror, **you will fail**.
6. If you run your business through Excel spreadsheets, **you will fail**.

FURTHER READING

Two seminal books that should be on every SaaS entrepreneur’s night table are:

The Lean Startup by Eric Ries. This book introduced many concepts that are now an intrinsic part of the language of the startup scene: pivot, minimal viable product, Build-Measure-Learn and continuous deployment.

Lean Analytics: Use Data to Build a Better Startup Faster by Alistair Croll and Benjamin Yoskovitz. This is a natural follow-on the ***The Lean Startup*** and has the quality of not just discussing metrics for startups, but how to turn those metrics in actionable decisions.

5. THE TECHNICAL CHALLENGES OF SAAS

"... multi-tenancy is much more of a marketing event than a technology event. Whoa! That sort of thing will get me excommunicated from the SaaS Church of Benioff. Well, I'm sorry, but it's true. Multi-tenancy is all about what we used to call 'green crystals marketing' at Borland ... When you're having a hard time differentiating, you find something unique and make it your green crystals. They provide a reason to believe why your offering is better even if they aren't the whole reason or even most of the reason."
- blogger Bob Warfield

Like any cloud service, SaaS has several technical challenges. Obviously, the service needs to be scalable to hundreds, thousands, perhaps millions of clients without a significant increase in response time, or hosting costs. It is often said that "true SaaS" must be multitenant. Are there situations where multitenancy is not the preferred solution? Is it preferable to host the infrastructure yourself, or use an IaaS or a PaaS solution? And with any shared infrastructure, how do you ensure that one client's data is not visible to another client of the service?

In this section, we will present and discuss the various tenancy models and their impact on scalability and costs. Other technical challenges that are common to most SaaS providers will also be discussed.

5.1 Tenancy options

The tenancy options for SaaS service touch on three components:

- Application computing **infrastructure**, which can be shared or dedicated;
- **Data separation**, which can be physical or logical separation;
- **Application versioning**, which can be single version or multiple versions.

Multitenancy refers to a software architecture where a single instance of software runs on a server and serves multiple tenants (25). However, each tenant has its own set of data that remains logically or physically isolated from the data that belongs to the other tenants.

Since elasticity of computing resources is an inherent characteristic of a SaaS, financial viability is heavily influenced by economies of scale and resource pooling. Therefore, multitenancy, with infrastructure sharing, is the dominant architecture model for SaaS Providers.

In a typical three-tier web application architecture, multitenancy can occur at any of three levels: the connectivity/presentation layer ("web layer"), the application code/business logic layer ("app layer") or the data management/data persistence/data storage layer ("database layer"), or any combination of the three. Multitenancy should not be confused with virtualization where a single server is transformed so that each tenant appears to have his own server in isolation, which is a way if sharing infrastructure resources.

Single Tenant Architecture

A single tenant architecture is a single instance of a software application, supporting one tenant, with dedicated compute resources. An illustration of a single tenant architecture follows. We note that each of the two tenants have servers dedicated to them at each of the three levels: web, app and database.

This architecture provides the most independence between tenants, as well as the most isolation between them as well. However, it will be more expensive than the multi-tenant architecture we will present afterwards since there is minimal sharing of infrastructure. It will also have higher operational cost because of the large number of “boxes” that need to be managed.

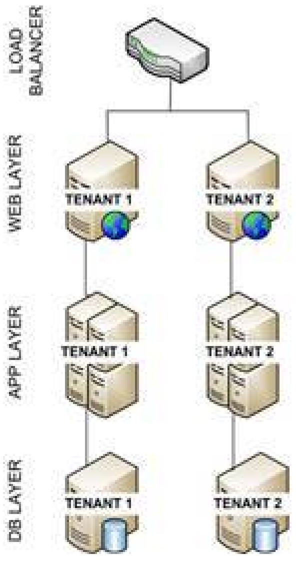


figure 16 – Single Tenant Architecture

As pointed out humorously in the quote at the beginning of the chapter, some proponents believe that SaaS and multitenancy are inseparable. However, there are some instances where single tenancy is the preferred way to meet the requirements of the business. As an example, I assisted one software company who decided to transform their on-premise business model to a SaaS business model. Their application was a Materials Resource Planning (MRP) solution; something that required very robust infrastructure to support this compute-intensive application. The satisfaction level of their on-premise installed base was low, but this was often the result of inadequate hardware choices made by their clients. Therefore, the move to SaaS was motivated by the desire to purchase and dedicate the appropriately sized hardware for each tenant. In a similar vein, they needed to isolate clients not just at a data level, but at the compute resource level so that a large MRP compute request from one tenant would not absorb all available capacity and negatively impact the other tenants. In this situation, a single tenant architecture made sense even though there was minimal infrastructure sharing, making it the most expensive solution.

Multitenancy at the Network and Web Layers

In almost all SaaS architectures, the networking layer (ISP links, firewalls, SSL decryption devices, load balancers) is shared since these devices are inherently designed to service multiple tenants. Even in the single tenant example above, normally the network layer would be shared.

Similarly, at the web layer itself, a web server such as Apache is designed to handle request for multiple URLs for different websites. Therefore, this layer is invariably shared as well. Therefore, in the following diagrams, we will omit the web layer for clarity, assuming that it is always designed as a multi-tenant layer.

Multitenancy at the App Layer.

In the following diagram, we illustrate multi tenancy at the App Server Layer. The App Server would be based on an Application server such as WildFly, TomCat, GlassFish, WebSphere, etc.

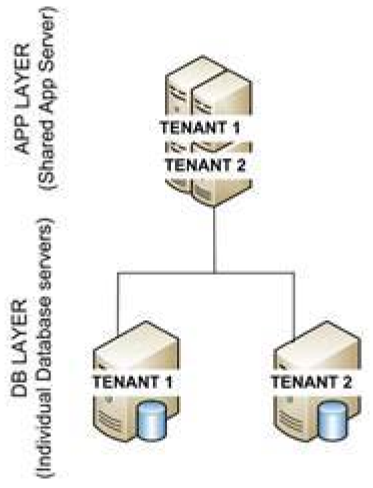


figure 17 – Shared App Layer Architecture

The Application Server is a critical component in multitenancy. Depending on the application design, each tenant may have dedicated application instance[7] or application instances may be assigned dynamically as required. If application instances are dedicated, this allows the SaaS Provider to support multiple versions of the software, if so desired.

Application versioning

In practically all situations, having all Tenants on a single version of the software is a best practice as it reduces complexity and effort for IT Operations, Support and Development Teams. However, for large enterprise clients, being able to control when upgrades are applied can be seen as an advantage, for example to respect "change freeze" periods that exist in certain industry sectors, or allow time for user training in large environments.

However, having a customized version for a specific client is a recipe for disaster for a SaaS Provider. As the number of clients grows, and the number of code versions increase, development and support costs will become unscalable. Differences in client requirements need to be implemented via configuration settings such that all clients run on the code base, but with different configuration settings, different configuration files, different CSS, etc.

Multitenancy at the Database Layer.

In the following diagrams, we illustrate two strategies for multitenancy at the Database layer. In figure 18, the database server is shared, but each tenant has its own schema. Therefore, the Tenant 1 app server will have the credentials to connect to the Tenant 1 database schema. Tenant 1 will not be able to connect the Tenant 2 schema, or any other tenant schema other than his own.

One of the challenges of a multitenant architecture at the database level is finding the right mix of Tenants to share the database server. The overall capacity of the server must be considered and care must be taken so that one Tenant does not grab most of the cycles, impacting performance for the other tenants. A very large tenant may need to be isolated, while several small tenants of similar size could share a database server with no issues.

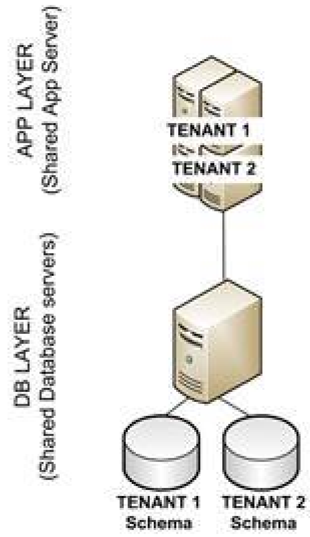


figure 18 – Multiple Database Schema Architecture

The other approach to database multitenancy is to combine all tenants into a single database. This configuration may imply a single code base for all tenants and enforces upgrade cycles. For this scheme to work, each database record needs to be prefixed with the Tenant ID to whom it belongs. It is then application logic which ensures that data privacy is maintained between clients. Alternatively, database security features such as Oracle UPD or MS/SQL Row-level Security can be brought into play.

This approach creates less infrastructure management overhead than the multiple schema architecture presented previously, but adds a storage and processing overhead. This is the approach taken by Salesforce, among others. Salesforce has developed a patented multi-tenant query optimizer to ensure that this overhead, however slight, is not an impediment to the scalability of the infrastructure.

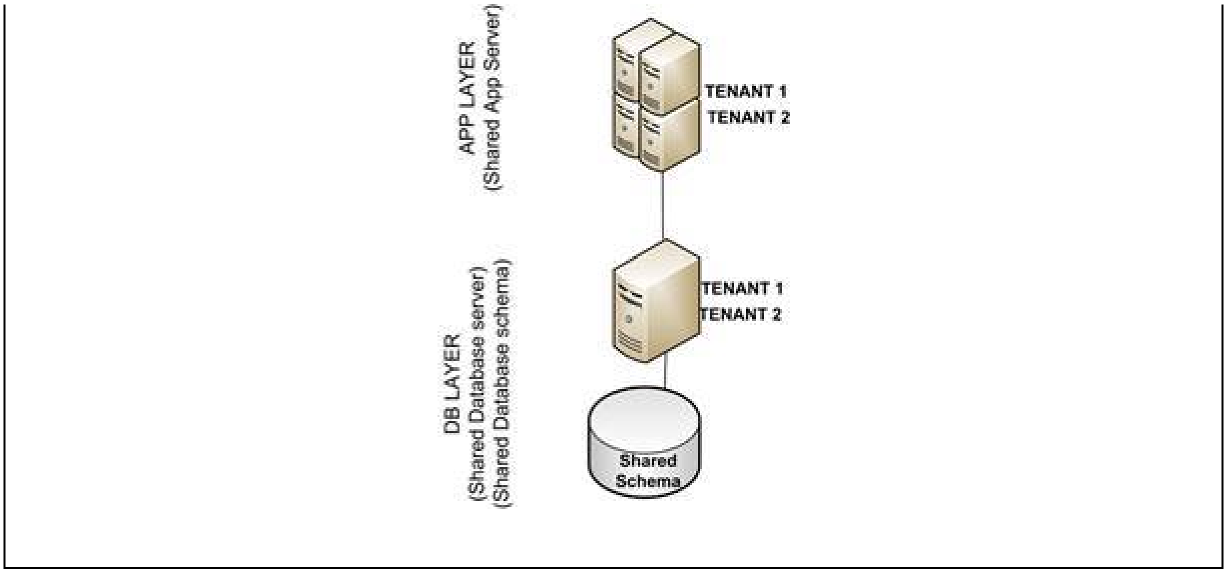


figure 19 – Single Database Schema Architecture

The above examples illustrate the key differences between a multitenant and a single tenant architecture. However, they do not illustrate scalability and high availability that are also requisite characteristics. We will discuss these in the following section.

5.2 Scalability

SaaS workloads are particularly sensitive to scaling issues due to their multi-tenant nature. To meet the requirement of infrastructure elasticity, the SaaS infrastructure needs to be designed to scale, be it vertically, horizontally or geographically.

Scaling vertically refers to adding (or removing) resources from a single node in the infrastructure. Typically, this would involve adding CPUs, adding memory, adding internal disks or replacing a server with a more powerful one.

Scaling horizontally refers to adding (or removing) nodes to the infrastructure. Typical examples would be:

- adding an additional web server to a web server cluster controlled by a load balancer;
- adding an additional app server to a cluster of app servers;
- adding a database server to a cluster of database servers (for example using the Oracle RAC architecture).

The development of low-cost servers and high speed interconnects such as Gigabit Ethernet and InfiniBand accelerated the adoption of this horizontal scaling strategy. Visit any colocation site and you will see examples of horizontal scaling done by adding dozens and dozens of 1U servers, known as a “pizza box” architecture.

For larger IaaS or SaaS Providers, scaling horizontally is not done on a server by server basis, but rather by adding a rack full of servers, or a “compute pod” consisting of a group of servers or racks of servers networked together as a unit.

For horizontal scaling to be effective, the system architecture needs to be loosely coupled with a stateless design. This can sometimes be a challenge when trying to port an existing on-premise product to become a SaaS service. However, when designing a new SaaS service, a stateless architecture is an imperative. In a stateless architecture, the service does not retain any information from previous service requests; this information will typically reside on the client.[8]

Having a stateless service allows the service to "scale out" with no changes to the application code itself. This has allowed cloud providers to introduce "auto-scaling" features in their service offerings, whereby additional capacity can be added automatically based upon a set of rules or policies. (26)

It should be noted that there is an overhead to adding nodes to a cluster, such that less than 100% of the processing capacity is made available to the cluster. If one server has a capacity of 1X, then a cluster of five servers will have a total capacity of less than 5X, due to the overhead of managing intra-cluster communications. The cluster overhead can severe when there is a requirement to synchronize application states, caches or databases. The impact is less severe for stateless applications. The exact mathematics of this cluster overhead is known as Amdahl's Law.

Geographic scaling refers to adding (or removing) entire data centers to the infrastructure with some static or dynamic way of distributing workloads among the data centers.

Optimization of a large-scale infrastructure running a single application, such as Google Search, is very different from tuning a workload that runs on a single server stack. The engineers at Google call this configuration: "***warehouse scale computer***" (27).

The typical architecture of a warehouse scale computer is:

- a single 1U server, or a single blade server;
- a 42U rack filled with 1U servers or blade enclosures, with an Ethernet switch;
- a cluster of 42U racks, networked together with a cluster level network aggregation layer.

This cluster "building block", representing many thousands of CPU cores can be replicated many times within a physical data center. By parallelizing workloads, since all servers are running the same application, contention within the warehouse scale computer will occur in the networking fabric or in the storage fabric, not on the backplane of a single 1U server. (28)

5.3 High Availability

Most clients of SaaS services have the expectation that the service will be "always on". We will discuss Service Level Agreements (SLAs) and contractual commitments in Chapter 8. However, consider that for a specific user, the service is either 100% available or 0% available when he is trying to get work done. An availability measurement of 99.99% has no real meaning when the system is unavailable and he has work to do[9].

There is an immediate and direct correlation between systems availability and customer satisfaction. There are others, but availability is a big component. Therefore, the SaaS Provider should design for high availability, the level of which will be determined by business, financial and technical considerations.

The other way to state this design requirement is to state that we should design for seamless recovery from failure. Scenarios should be gamed-out to plan for the failure of a server, a storage unit, a network segment, or even an entire data center.

Chaos Monkey (29)

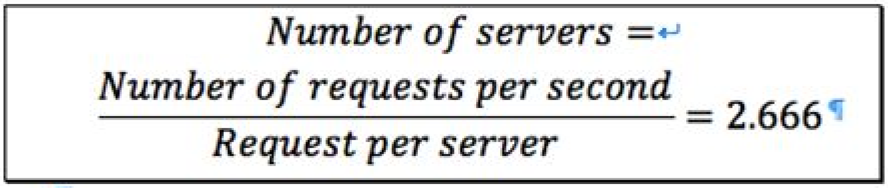
Netflix, who run their video streaming service on the AWS IaaS service, have taken "planning for failure" to the next level. The Chaos Monkey randomly introduces faults into the service, in production! Since the faults are random, the designers of the Netflix application need to build fault-recovery or mitigation mechanisms into the application, so that the end-user experience is unaffected.

The introduction of the Chaos Monkey reinforced a culture of building systems that can automatically recover from random faults, with no human intervention.

In practice, designing for high availability is a lot easier with stateless applications. Applications with persisted data is a challenge for High Availability as data must be copied/replicated, presenting a scalability issue.

High Availability designs will use the same horizontal and geographic scaling techniques we discussed earlier, but with added capacity to recovery seamlessly from a failure.

For example, if your SaaS Service experiences a daily peak of 400 requests per second. These requests are processed by a load-balanced cluster of servers; each server can handle a peak load of 150 requests per second. Therefore, you will need to dedicate three servers to this service:



$$\mathit{Number\ of\ servers} = \frac{\mathit{Number\ of\ requests\ per\ second}}{\mathit{Request\ per\ server}} = 2.666$$

However, if one of the servers fails, what happens to the service? The two remaining servers can only handle 2 * 150 = 300 requests per second, which is far below the peak load of 400.

Therefore, to make this configuration highly available, we should install four servers. So, if we lose a server to failure, the remaining three servers will still have the capacity of meeting the peak load. In this example, a design with three servers would be known as an N configuration. Four servers would be an N+1 configuration, while five servers would N+2.

The same logic needs to be applied when designing database server clusters, networking infrastructure and auxiliary services. But what happens if we lose an entire data center? We will discuss that scenario in section 7.7 dealing with Disaster Recovery.

It should also be noted that high availability is not just a technical challenge. Achieving Five Nines of availability is more than just calculating the correct number of servers. Other elements that contribute to operational excellence and great availability numbers are:

- Standard operating procedures, especially when multiple teams, possibly geographically separated, are managing the service;
- Rigorous change control procedures;
- An efficient and effective QA or DevOps group, along with Test Driven Development;
- Infrastructure Hardening on a continuous basis;
- Root Cause Analysis with actionable Lessons Learned;
- Monitoring and Alerting;
- Capacity Planning;
- Rigorous Release Management processes.

5.4 Hosting options

There are three hosting options available to a SaaS Provider:

1) Self-hosted; that is building out a data center, installing racks and servers and operating everything from A to Z;
2) Colocation; essentially obtaining a "condo" within a colocation facility that provides a dedicated space but shared power & cooling infrastructures;
3) Cloud: leveraging an IaaS or PaaS supplier.

With the wide range of IaaS, PaaS and Colocation offerings, option 1 is rarely the most cost-effective or viable choice, except for the largest organizations, such as a Facebook, who have sufficient scale on their own. Considering that most startups are severely cash-constrained and are trying to limit CAPEX spending to avoid being stuck in the cash-flow trough, most choose Option 3, at least to start.

The following chart, from the 2016 Pacific Crest Private SaaS Company Survey (21), shows that indeed Option 3 (IaaS and PaaS) is the preferred hosting option, with AWS being the dominant supplier. Consider **self-hosted** in this chart to include both our Option 1 (self-hosted) and Option 2 (colocation) and represents only a third of the SaaS Providers surveyed.

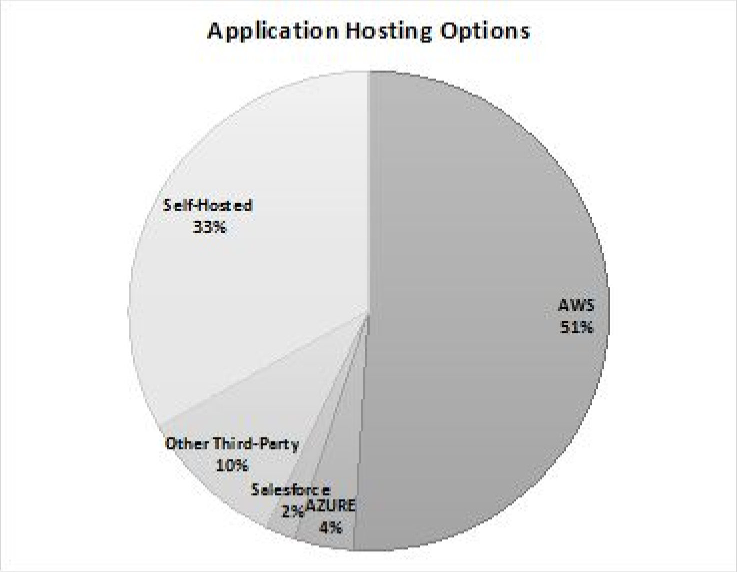
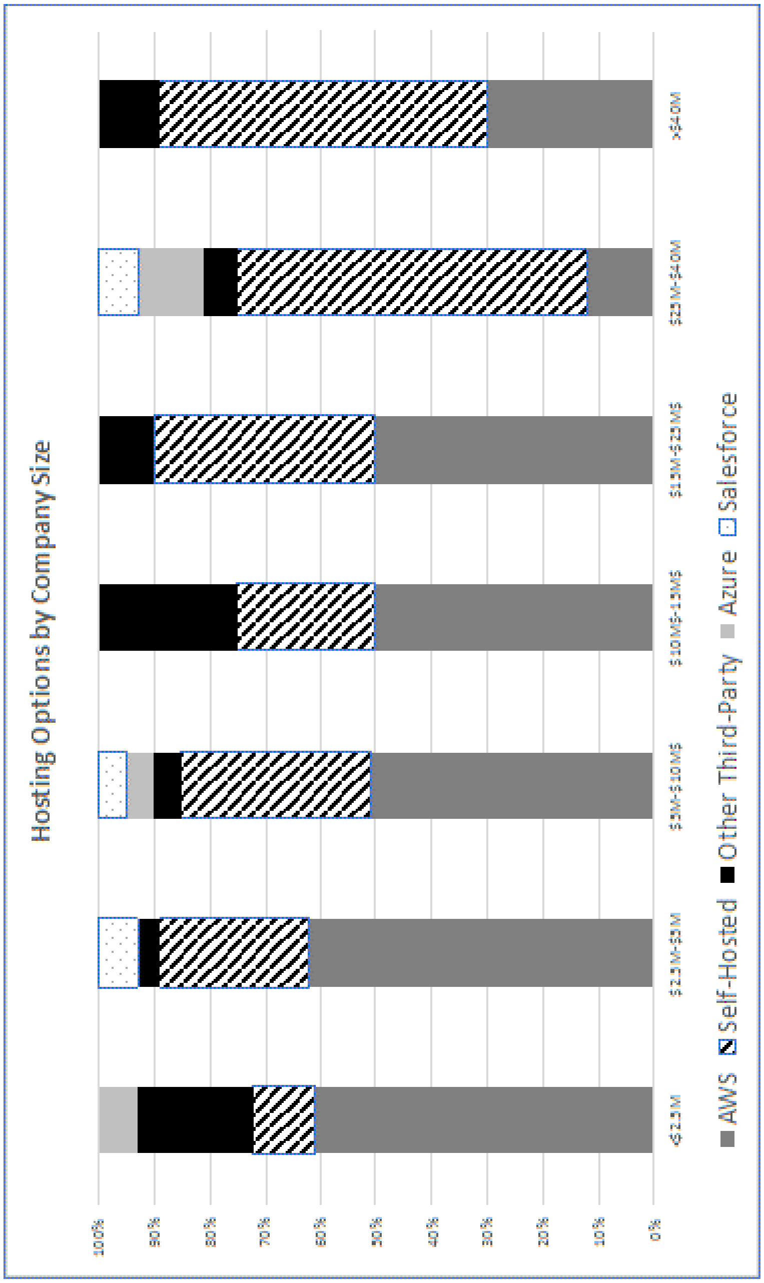


figure 20 – SaaS Application Hosting Options

However, if we breakdown these numbers by size of the SaaS Provider, some interesting trends emerge (see figure 21).

As we can see in the breakdown by company size, 89% of startups (companies with less than $2.5M ARR) will utilize a IaaS/PaaS solutions. In this category, AWS is the preferred hosting option for 61%.

When SaaS revenues reach $5M, AWS as a hosting option falls to 50%. At $25M to $40M ARR, AWS drops to 12%. At that point, self-hosting becomes the preferred option for 63% of SaaS Providers. The switch to self-hosting comes with increasing IaaS/PaaS fees and the economies of scale that are possible with self-hosting at that scale. The story of Dropbox (see text box on the following page) is quite eloquent in that regard.



The story of Dropbox's move out of the cloud. (30)

Dropbox's journey is an interesting one. We often assume that it is difficult to beat the economies of scale of the large hosting providers like Amazon and Google. So why would Dropbox choose to move off Amazon's cloud?

For the first eight years of its existence, Dropbox utilized AWS' S3 storage service. It would store the files Dropbox users wanted to share in Amazon's cloud, while it stored the meta-data required to retrieve those files on Dropbox servers in a Dropbox data center. So, when a user stored a file on Dropbox, it was actually stored on an Amazon server in an Amazon data center.

This strategy allowed Dropbox to scale to over 500 million users with limited capital spending on infrastructure. Dropbox then designed and built out its own storage platform and by 2016 had moved 90% of their user's files out of Amazon S3 and into Dropbox's own storage. What was the logic?

First, Dropbox had achieved a scale where their Amazon costs had become a significant portion of their operating costs. As Aditya Agarwal, Dropbox's VP of engineering, says: "Nobody is running a cloud business as a charity." Secondly, Dropbox had the resources to engineer their own storage platform, incorporating into it "Dropbox-only" features to distance itself from competitors like Box and Google Drive.

So, cost and features drove Dropbox to build their own storage cloud. However, few companies have the scale and the resources to make this a worthwhile effort. "The right answer is to actually not do this yourself," confirms a Dropbox engineer.

When adopting an IaaS or a PaaS solution, one must look at the services offered and transpose the generic multi-tenant architectures, (presented in section 5.1) in an instantiation for that cloud solution. For example, Amazon presents the different tenant isolation schemes that are possible within the AWS service (31):

- Model #1 – Tenant isolation at the AWS account layer;
- Model #2 – Tenant isolation at the Amazon VPC layer;
- Model #3 – Tenant isolation at Amazon VPC Subnet layer;

- Model #4 – Tenant isolation at the Container Layer;
- Model #5 – Tenant isolation at the Application Layer.

Each model has its advantages and disadvantages for cost, manageability, monitoring, security, billing, etc.

5.5 Connectivity Options

Another important technical design choice is the connectivity to the Internet. If you choose an IaaS or PaaS solution, then you delegate this “problem” to your hosting provider. AWS, Google, Azure or Salesforce will monitor and optimize Internet connectivity to their hosting locations. Despite this, you may still have users of your service, because of their geographic location and the nature of their local ISP, that will experience latency issues.

By moving to a self-hosted option, you will have the most connectivity possibilities available to you. Yes, you can just order one large dumb pipe from one ISP. Or you can do a better job by ordering two pipes from two providers, to have some level of redundancy, and load-balance the traffic between the two links using a BGP protocol.

The next step in sophistication is to use an intelligent routing device or service. One example of this type of service is Internap’s Performance IP service[10] (32). This service, if available in the data center where you are hosting, will provide connectivity to a dozen or more of the most important ISPs. When a request from a user comes in from a specific network, essentially the user’s ISP, then the Performance IP service will ensure to place the response directly on the same ISP network, saving one or more hops, and reducing latency.

Another service can that can alleviate latency issues is the use of a Content Distribution Network (CDN). A CDN is a globally distributed network of proxy servers located in data centers throughout the world. These proxy servers allow the SaaS Provider to place content (static web objects, downloadable objects, streaming media, etc.) as close as possible to the end-user. This can significantly reduce the amount of wide-area-network traffic needed to deliver a transaction to the end user’s device.

Some of the better known CDNs are Akamai, Level 3, Limelight and Cloudflare. However, most major telco’s and Cloud Suppliers (Azure, Amazon, Rackspace, etc.) all have their own CDN offerings.

Another option is to acquire a private link to the public Cloud supplier that you are using. For example, Microsoft Azure offers ExpressRoute to enable private connections to its data centers. The AWS equivalent is Direct Connect and Google's is Direct Peering. These direct connection services establish a private connection to the Cloud Provider's hosting location. These will typically use a connection provided by a third-party network supplier. You should ensure that encryption will be applied to data in transit over this connection.

5.6 Other technical challenges

Some other common technical challenges encountered in SaaS solutions are reporting, integration and orchestration.

Reporting. It is important for the SaaS Provider to consider the reporting requirements of his Tenants. A feature-rich reporting solution is often required since the Tenant cannot interface directly with the database like he could in an on-premise installation. Reporting can also be a pain point for the SaaS Provider, especially if the SaaS Tenant has the capability of customizing and building his own reporting queries. A badly written report can generate many "full table scans" and bring a database server to its knees, especially if the user can run his report at any time during the day, even in peak hours.

There are several ways of mitigating the impact of reporting, although some of these choices might not be appropriate in certain business situations:

- Only offering predefined reports that have been designed and tested by the SaaS Provider;
- Restrict the number of rows that can be read to generate the report; schedule long-running reports for off-peak hours or over-night;
- Performing a nightly copy or a asynchronously replicated read-only copy of the production database to a reporting database or a data warehouse that is used solely for reporting with no performance SLAs.

Integration. SaaS Tenants will want to have ways of loading data into the SaaS service, sometimes in bulk. And likewise, some Tenants will be looking for ways to extract SaaS data, to feed their internal systems. Depending on the quantity of data being manipulated, the same performance issues as reporting can occur.

However, a bigger issue is understanding the data model behind the SaaS Application. For example, data uploads need to respect data formats and referential integrity constraints of the database. SaaS Providers will often be reluctant to share their data model, which they may consider as Intellectual Property (IP) to be protected. In some cases, SaaS Providers will force clients to use their Professional Services staff to build the integration scripts to protect their IP and ensure the quality of the integration scripts.

From a tools perspective, Enterprise solutions may use integration tools such as Tibco, WebMethods or a SaaS solution such as Boomi or Zapier. For simpler integration patterns, the SaaS Provider may offer several Web Services or APIs as part of the standard SaaS offering.

SaaS Tenants should be especially cautious if their business requirements call for implementing numerous integration feeds with back-end systems. These integrations can be costly and time consuming to set up initially. They can also generate massive potential for Vendor Lock-In over time.

We are seeing an even greater shift towards APIs as many SaaS offering are being built and marketed as “screenless software”. Since these are proprietary APIs, users must be conscious that there is the potential for vendor lock-in. Some examples of common SaaS APIs today are:

API	PROVIDER
Mapping	Google
Payments	Stripe
SSO	Auth0
AI	IBM Watson
Communications	Twillio
Background Check	Checkr
Healthcare	Human API

Orchestration. Many of the early successes in SaaS were stand-alone products that required little or no integration. Over time, integrations have become more complex and time dependent.

I worked at a SaaS Provider providing a recruiting solution. One of our Tenants was a trucking company who hired drivers on a contract basis, per long-haul trip. When the dispatcher pressed the “hire” button, this initiated an integration into an SAP HR system which generated an employee number which was required by the dispatch system and insurance reporting to release the truck. If anything broke in our system or the subsequent integration points, then the driver could not leave the yard with the load until it was resolved.

This example illustrates some of the challenges that occur when a SaaS system is logically dependent on up-stream and down-stream systems. There is limited tooling that can monitor the end-to-end transactions since the SaaS Provider has no access to the outside systems and the SaaS Tenant has limited visibility within the SaaS infrastructure.

These challenges will only grow with the push to Services-based architectures and micro-services.

5.7 Upgrades

The SaaS Provider is responsible for all actions relating to infrastructure upgrades and version upgrades in the application itself. The SaaS Provider needs to establish a strategy to handle this, with as little impact as possible on the user base, and make sure the requisite conditions are understood by the Development, QA, DevOps and IT Operations teams. These conditions need to be reflected in contractual documents and policies so that SaaS Tenants are also fully aware of how upgrades will be applied.

Among the policy decisions that the SaaS Provider needs to make are:

- Do we establish a fixed maintenance window for maintenance activities?
  Is a maintenance period required for infrastructure changes? How long does the "typical" version upgrade take to complete (allowing time for a rollback if required)?
  When is the maintenance window for our clients that have global operations?
- How do we manage the upgrade calendar?
  Do clients get to choose when they upgrade? How much notice do they get? Do they get access to a trial site for testing before the actual upgrade in Production? Do we establish a fixed calendar (example: Spring and Winter, like Salesforce) or we align with the Development Roadmap?
- How do we actually execute the upgrades?
  Do we upgrade all tenants at once or is it a phased approach? or Hosting Location by Hosting Location? Do upgrades involve only code changes or is there the possibility that there are database changes that require more downtime to complete? How much automation can we apply to the process? What happens if an upgrade fails to complete?

As we have seen, single tenant architecture and some multi-tenant architectures allow for tenant by tenant upgrades. This can fast become a painful process to maintain and there is always the danger that a Tenant always finds a good reason to delay his upgrade, with the result that the SaaS Provider will have to support a Tenant who is several versions behind.

The SaaS Provider may wish to impose fixed date upgrades for all Tenants, and indeed, there may be no alternative for true multitenant architectures. In this case, Tenant communications and adequate testing (to limit any roll-backs) are key.

5.8 Performance & penetration testing

The SaaS Provider will need to devise a strategy to do performance testing and penetration testing. The challenge here is that if these tests are performed against the production environment and a weakness is uncovered, the test could potentially impact guaranteed service levels for Tenants.

On the other hand, performing these tests in a test, development or staging infrastructure that is not exactly like the production infrastructure leaves a margin of error that may or may not be acceptable. While performance testing can be done in these other environments, a penetration test is only valid if it is done against the Production infrastructure.

Due to the possible impact on service levels, SaaS Providers invariably add a clause to their contracts stating that performance and penetration tests cannot be done by Tenants. There is usually a provision allowing a tenant to have access to the results of the tests performed by the SaaS Provider, with the proviso that the Tenant cannot publish or share the results.

Ideally, the SaaS Provider will want to do performance testing, before a release, with real workloads and real data in a QA or DevOps environment. Several tools exist that can simulate user interactions and generate load against a test system. The challenge then becomes access to test data. Testing an application against a small database might not reveal inefficiencies, like a full table scan, that would show up with a large database. Rule of thumb is to do performance testing against a database that is at least as large as your largest database in Production.

In theory, the SaaS Provider's staff should not have access to SaaS Tenant data, or only under exceptional conditions. Therefore, copying a SaaS Tenant's database to the QA environment should be a no-no. One possible way to get around this restriction is to depersonalize the information in the database before copying it (i.e. striping out/changing fields that contain names, addresses, account numbers, credit card numbers, etc.). However, in some cases, even this might not be possible and the SaaS Provider will have to populate a test database himself.

The SaaS Provider has no control over how his Tenants actually use his application. During my tenure at one SaaS Provider, the Development and QA teams tested against a depersonalized copy of our largest database in Production. However, when we upgraded to the new version in Production, we had many clients complain about sluggish performance. Why?

We discovered that the clients who were complaining were using a specific feature that our largest client was not using. This feature was populating a table in the database that was almost empty in the case of our test database. Adding an index on that table solved our performance issues.

5.9 Monitoring

Monitoring is a usual activity for an IT Operations group. The IT Operations group will typically instrument the IT infrastructure to detect abnormal usage levels, system alerts and network failures. The focus has traditionally been on “reactive monitoring”; an approach often called “break-fix”.

In addition to typical infrastructure monitoring, there are many tools that ping a service from an external site to detect a response and confirm that the site is running correctly. If there is an error or no response, then we can conclude that there has been a service disruption. This is also reactive monitoring. These tools will test the site at a fixed interval, say every five minutes, or in some cases every minute. This delay between tests means that, in some cases, the end-user will detect the service disruption before the monitoring has raised an alarm.

However, the expectation of the SaaS Tenant is that the SaaS service will be “always on” and “always available”. This means that the IT Operations needs to shift its focus to Proactive monitoring so that it cannot only detect failures, but predict failures before they have an impact on service levels.

This is a much more difficult challenge. It means instrumenting the service and determining patterns that represent “normal” behavior. This can translate into technical monitoring of performance, throughput, response times and error rates. Business activity monitoring can also be a good indicator of a system behavior; metrics such as number of unique visitors, number of functional transactions, number of errors or abandoned sessions, etc.

5.10 Technical debt

A SaaS infrastructure is rarely static; there are almost constant changes to the infrastructure, the application, the number of tenants, the storage and compute requirements, etc.

Like any system, entropy increases over time. As a consequence, availability and performance will decrease will time, unless investments are made to keep the system performing adequately.

> “Software systems go through a predictable lifecycle starting with small well-crafted solutions fully understood by a single person, through the rapid growth into a monolith of technical debt, thus fissioning into an ad hoc collection of fragile services.”
> – Lee Atchison, “Architecting for scale” (33)

Availability and performance can decrease over time due to:

- Resource exhaustion;
- Unplanned load changes (bursts in business activity, intensive reporting, intensive integration tasks, web crawling, etc.);
- Increased number of “moving parts”;
- Outside dependencies;
- Technical debt within the application.

A real-life example: Performance was especially sluggish on a set of servers in a SaaS Applicant Tracking System. The IT Operations team searched the logs for some technical reason for the slowness. The simple answer was a major healthcare provider ran a one-page ad in the New York Times to hire thousands of nurses. Understanding (and being forewarned about) this unusual business activity would have allowed the IT Ops team to prepare for this increase in load. The alternative is to allow the infrastructure to “auto-scale” based on transaction volumes.

Another example: The Development Team, in an effort to increase performance, started to pre-load data into the app server upon user login. Over time, more and more data was being preloaded. However, no one in the Development team was responsible for removing any “pre-loads” that were no longer critical; indeed, the Development staff no longer knew what most of the pre-loads were for nor the effect of removing them. When complaints started coming in regarding the long time to complete the login, tests revealed that over 150,000 lines in the database were being read and uploaded before every login. A good idea gone bad.

To avoid this type of technical debt, it is important that all internal stakeholders (Product Management, Sales, Development, DevOps, IT Operations) understand the Product roadmap and ideally, have some input in the planning process. It is important that some space is left in every release to invest in technical or operational improvements.

6. SAAS SERVICE DELIVERY

The customer's perception is your reality. – Kate Zabriskie

IT Service Delivery is not a new topic. Several frameworks have been developed to describe a standard set of service delivery processes. The ITIL model (34) and COBIT (35) being two popular frameworks. However, in the world of SaaS, what is different from operating an on-premise solution?

In this section, we will present a SaaS delivery framework, highlighting the key processes and especially, the key differences with on premise software delivery.

6.1 The ITIL model

ITIL is an acronym for Information Technology Infrastructure Library, is a set of practices for IT service management (ITSM) that focuses on aligning IT services with the needs of business. In its current form (known as ITIL V3), ITIL is published as a series of five core volumes, each of which covers a different ITSM lifecycle stage. The ITIL framework also underpins the international standard for Service Management ISO/IEC 20000.

Three of the core ITIL volumes: Service Design, Service Transition and Service Delivery have their place in a SaaS world, but should be adapted to the specific context of this delivery model.

6.2 ITIL Service Design

ITIL Service Design presents the framework for the design of IT Services, processes and other aspect of Service management. Within ITIL, design is not limited to technical design, but encompasses all elements that are relevant to service delivery. Service Design should be thought of in broad terms to encompass all elements relevant to technology service delivery.

The key processes of ITIL Service Design are:

1. Service Catalogue Management
2. Service Level Management
3. Capacity Management
4. Availability Management
5. IT Service Continuity Management
6. Information Security Management
7. Supplier Management

Service Catalogue Management provides and maintains accurate information on all services being provided by the IT Operations group.

In a SaaS organization, the Service Catalogue should be a collaborative effort between Product Management, DevOps and IT Operations. In addition to describing the functional characteristics of the service, attention should be given to delimiting what configuration and operational actions are to be performed by the SaaS Provider and which ones are the responsibility of the SaaS Tenant in a self-service mode. The SaaS Provider should also document which services are part of the base service, and which services involve an additional charge. For example, is restoring a database from a backup included in the base service or does it generate a service charge?

Service Level Management negotiates, agrees to and documents appropriate IT service targets. It then monitors and reports on the IT Operations ability to meet those service targets.

Obviously, Service Level Management needs to be in sync with contractual obligations, however, there may be more stringent internal “stretch goals”. Service Level Management can be further complicated, over time, by different versions of the Service Level Agreement, with different SLA targets, in different contracts.

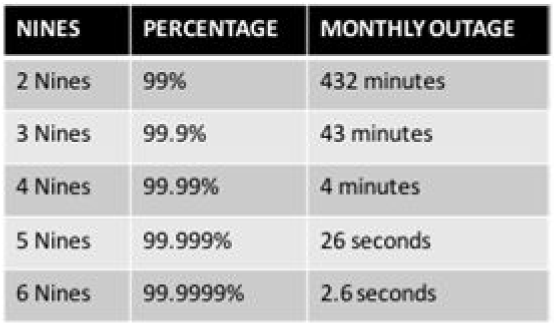
Capacity Management ensures that appropriate IT capacity exists and is matched to the current and future needs of the service, in a timely manner.

Capacity Management is crucial to maintaining the “infinite elasticity” of the SaaS service. Time to react is relatively short if the SaaS service operates on an IaaS service. However, if the SaaS service is self-hosted, the delay to order, install and configure new equipment can add up to several months. Capacity planning will need to consider organic growth, new capacity requirements of the new versions being rolled out and the Sales pipeline for the coming months.

Availability Management aims to ensure that the level of service availability is matched to or exceeds the current and future needs of the business, in a cost-effective manner.

A distinction needs to be made between reliability and availability. Reliability generally refers to the ability of the system to perform to specification, and is one component of Availability. Availability is determined by reliability, maintainability, serviceability, performance and security.

Availability is usually calculated as a percentage and availability target are often expressed as a “number of nines” calculated over a period of a month or a year. The figure below translates “the Nines” into minutes of downtime per month.



NINES	PERCENTAGE	MONTHLY OUTAGE
2 Nines	99%	432 minutes
3 Nines	99.9%	43 minutes
4 Nines	99.99%	4 minutes
5 Nines	99.999%	26 seconds
6 Nines	99.9999%	2.6 seconds

figure 22 – The Nines

The goal of **IT Service Continuity Management** is to support the overall Business Continuity Management process by ensuring that the required IT technical and service facilities can resumed within required, and agreed to, business timescales.

For a SaaS startup, developing a Business Continuity Plan (BCP) can be a challenge. Cash flow and pricing challenges will force many startups to push to question out until later. When selling into the Enterprise space, knowledgeable clients will insist on a credible BCP. As the SaaS Provider's business grows, the Executive Team or the Board of Directors will become increasingly aware that the entire business could be put at risk by a major disaster.

The **Information Security Management** process aligns IT security with business security and ensures that information security is effectively managed in all service and service management activities.

As we mentioned in previous chapters, one of the primary concerns of SaaS Tenants is the security of their data. Since this is such an important topic, we dedicate an entire chapter to the subject – **Chapter 7 SaaS GRC and Security**.

The goal of the **Supplier Management** process is to manage suppliers and the services they supply.

For IT Operations, key suppliers will obviously include hardware and software suppliers, IaaS or PaaS suppliers, colocation suppliers, ISP and

CDN providers, etc. For the SaaS Provider, one important element of Supplier Management is to ensure that the SaaS Provider's obligations (for security, availability, etc.) are reflected in each of his supplier's contracts.

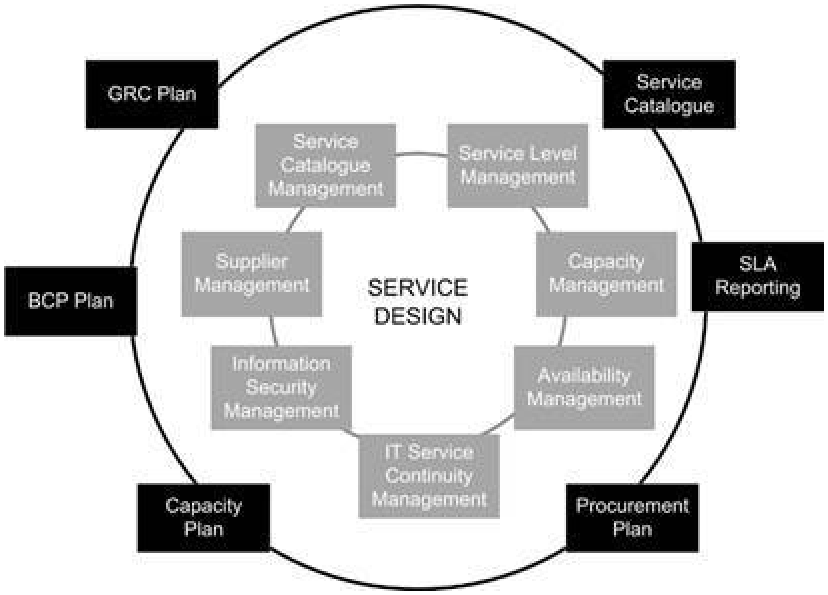


figure 23 - Service Design Processes and key outputs

6.3 ITIL Service Transition

ITIL Service Transition presents the framework for transitioning new and changed services into operations. While ITIL Service Operations, presented in the next section, aims at stability, Service Transition aims at change, while managing the risks of failure and disruption due to those changes.

The key processes of ITIL Service Transition are:

1. Change Management
2. Service Asset and configuration management
3. Release and deployment management
4. Service validation and testing
5. Change evaluation
6. Knowledge Management.

The **Change Management** process aims to enable beneficial changes to be made, with a minimum disruption to IT services.

Be aware that "traditional" Change Management processes can be challenged by DevOps processes with continuous integration and continuous delivery. There is always benefit to having standardized methods and procedures for changes, however, there is always be a certain amount of tension between process rigor and speed of execution.

The purpose of **Service Asset and configuration management** is to identify, control, record, report, audit and verify service assets through the complete service lifecycle.

In ITIL's model, all this asset and configuration information resides in an integrated Configuration Management Database (CMDB). However, in most IT organizations, this information is segmented and stored in a variety of point solutions and information silos. Proper management of this information is a key challenge but the payback can be significant in terms of organizational efficiency.

The objective of **Release and Deployment Management** is to plan, schedule and control the movement of software releases to test and production environments.

In a SaaS environment, this needs to be highly automated to minimize downtime and client disruption. The process needs to cover all instances, including staging and QA environments. Oversimplifying perhaps, there are three main strategies used by SaaS Providers:

- Release and upgrade as often as possible or required; this is the strategy used by Atlassian for their Confluence product: feature releases come out every three to four months, while bug fix releases come out every three to four weeks;
- Release at fixed dates according to a pre-announced schedule; this is the Salesforce.com strategy who release a new version seasonally in Spring and Winter every year; Workday have also adopted a “two releases per year” strategy;
- Release feature releases according to the developing testing time required; so, no rigid calendar. See for example ServiceNow’s release cycle.

The purpose of **Service Validation and Testing** is to implement a structured validation and test process that will quality assure a release and identify any issues, errors, or risks throughout Service Transition.

There are multiple ways to implement an effective test process and organization. Likewise, there are multiple types of tests that may be appropriate in each business situation: usability testing, accessibility testing, documentation testing, security testing, deployment & migration testing, operability testing, performance testing, etc.

Change Evaluation considers whether a service change will deliver the business value anticipated and whether the organization should proceed with the change.

An effective Change Evaluation process should be much more than a simple Go/No-Go decision process. It should also evaluate the intended effects of a service change, as well as the unintended effects, all in the context of determining “value to business”.

The primary purpose of **Knowledge Management** is to improve efficiency by reducing the need to rediscover knowledge. This process ensures that the right information is delivered to the appropriate place or competent person at the right time to enable an informed decision.

There are many tools that can assist in building a Service Knowledge Management Systems including wikis, document repositories, and enterprise search tools such as Coveo. In addition to tools, several process activities are required for effective knowledge capture and knowledge transfer.

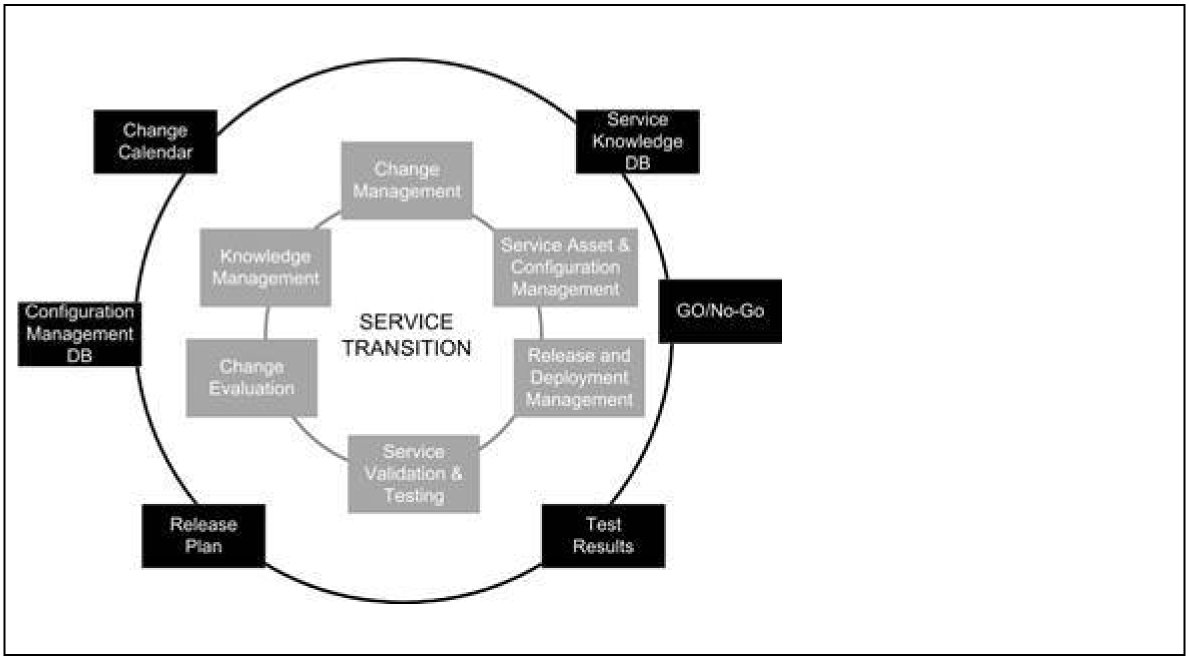


figure 24 – Service Transition Processes and key outputs

6.4 ITIL Service Operation

ITIL Service Operation provides guidance on achieving effectiveness and efficiency in the delivery and support of IT services to ensure value for the customer and the service provider. ITIL Service Operations aims to maintain the stability needed to meet agreed levels of service while allowing for changes in design, scale, scope and service levels.

ITIL Service Operations can be organized according to two different control perspectives: reactive and proactive. The proactive perspective often coming with more advanced maturity levels and more significant investments in tools and automation.

The key processes of ITIL Service Operation are:

1. Event Management;
2. Incident Management
3. Request Fulfillment
4. Problem Management
5. Access Management;

Event Management is the process that monitors all events that occur through the IT infrastructure to allow for normal operation and detect and escalate exception conditions.

Effective monitoring, by the SaaS Provider, is one of the potential strengths of the SaaS model. In addition to monitoring infrastructure and software components, the design of this process will require an understanding what constitutes normal versus unusual behavior, as well as a process to correlate events. Tools for logging events and sending notifications are needed to support this process.

Incident Management aims to return the service to an operational status as quickly as possible. In ITIL terminology, an incident is defined as an unplanned interruption to an IT service or reduction in the quality of an IT service.

Incidents can be signaled by Event Management (monitoring), a support call, chat or email to the Service Desk or a user message from a Web interface. Incidents need to prioritized according to their urgency and their impact. A specific process is usually put in place for “major incidents”.

Request Fulfillment is the process of dealing with Service Requests from users. This process is limited to those requests for standard services for which a pre-defined approval and qualification process exists. These are mostly low risk, high frequency, low cost changes that do not warrant invoking the complete Change Management process. Examples of standard requests could be creating a new user, a password reset, etc.

The primary objectives of **Problem Management** are to prevent problems and resulting incidents from happening, to eliminate recurring incidents and to minimize the impact of incidents that cannot be prevented.

ITIL defines a “problem” as the cause of one or more incidents. Thus, Problem Management includes the activities for root cause analysis and ensuring the appropriate resolutions and workarounds are implemented. Problem Management can be a reactive process, linked to Incident Management, but can also be a proactive process, part of continual service improvement.

Access Management is the process of granting authorized users the right to use a service, while preventing access to non-authorized users.

Access Management can be initiated by a Service Request through the Service Desk. However, most SaaS Providers will adopt a more “low-touch” approach, either allowing the SaaS Tenant’s administrators to self-managed access rights through an admin portal, or in the case of an Enterprise service, using a Single-SignOn (SSO) service.

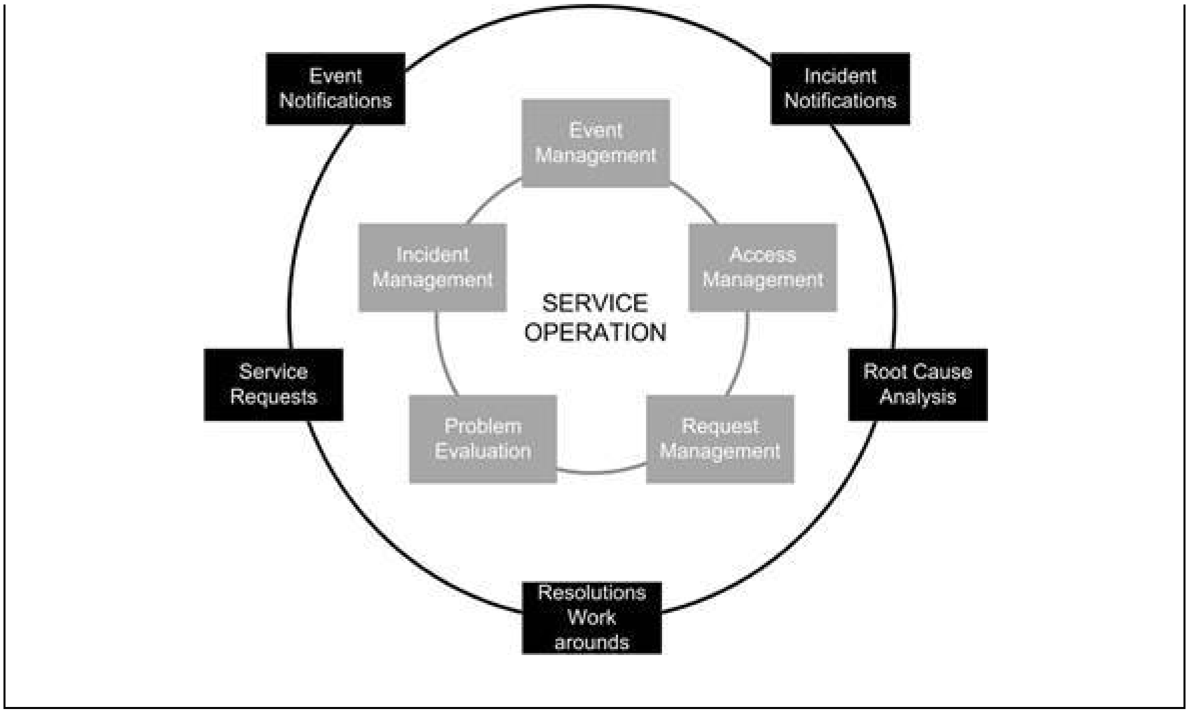


figure 25 – Service Operation Processes and key outputs

6.5 Key SaaS transparency metrics

At the risk of repeating ourselves, a SaaS Tenant's trust in the SaaS Provider and the service provided will influence renewal rates and upsells. One way to positively influence trust is for the SaaS Provider to be transparent and forthcoming regarding the service levels delivered.

There are several key metrics that the SaaS Provider should strive to provide on a regular basis (monthly or quarterly) to his Tenants, even though there might not be a contractual SLA tied to the metric. Certainly, a SaaS startup could start by collecting and publishing these metrics internally for a certain time before making them available to his client base.

Availability. This is metric #1 for both Providers and Tenants. The SaaS Provider should make a "Trust Page" available that shows the current operational status of key services. It should also display historical availability statistics, to be shown as a daily percentage and a monthly/quarterly/yearly average.

There are several SaaS-based offerings to host a "trust site". A SaaS-based solution should be considered since hosting your own trust page on the same infrastructure as your production SaaS Service can create a "single point of failure". If your production SaaS Service is having issues, you do not want your trust site to suffer from the same issues and limit your ability to inform your Tenants.

Outages. The number and duration of outages should be reported. A root-cause analysis should be provided for major outages, within a reasonable timeframe.

Usage. This metric should be mandatory if there is a usage-based component to the pricing model. Ideally, these should be functional or business metrics, although technical metrics should be collected, analyzed and monitored by the IT Operations team. For SaaS Provider management and SaaS Tenant stakeholders, it is more useful to know that

6 job posting were filled this month on our Applicant Tracking System, than to know that there were 5.7 billion IOps on database server ORA9587.

Business-related usage statistics help the SaaS Tenant analyze usage and adoption patterns. Reports also help SaaS Tenants and SaaS Providers understand the value delivered and how to make future investment decisions. Low usage levels of a certain module can indicate that it is not delivering the expected value, or perhaps additional training is required.

Performance. Response time measurements can also be used as an indication of the health of the service. Although it should be clear what is being measured and how it is being measured. For example: is the response time being measured for actual user transactions or a synthetic transaction designed specifically to measure performance?

Does the response time include network delays? Is the synthetic transaction being sent from a site not far from the data center? Are the statistics representative of the real user experience? (i.e. a user in Australia will have very different response time from a user in New York when the hosting location is in New Jersey.)

Security breach notifications. There should be a formal process in place for disclosure of any security breach that could potentially affect the Tenant's data. The SaaS contract, that we will discuss in Chapter 8, should specify any liability relating to security breaches, such as cost of customer notifications or fines.

While we are on the topic on SaaS Tenants communications, just because you don't hear from your clients, that doesn't mean that all is well. Some clients will be very vocal about dissatisfaction regarding the service, while others will quietly transition to a new Service Provider with little advance warning.

Therefore, the SaaS Provider should always be proactive in monitoring usage levels and adoption. Quarterly, semi-annual, or annual business reviews, depending on the importance of the client, should be planned systematically. Build a Customer Success organization, compensated on renewals, that is separate from the Sales organization. And finally, take the time and effort to measure customer satisfaction, through customer satisfaction surveys or net promoter scores.

7. SaaS GRC and SECURITY

We will bankrupt ourselves in the vain search for absolute security.
- Dwight D. Eisenhower

In the Due Diligence process for purchasing Enterprise SaaS applications, security is often a major topic. The SaaS Tenant wants to ensure that his data will be available, reliable, and protected from unauthorized access. The SaaS Provider wants to be able to demonstrate to prospective and renewing subscribers that his software, infrastructure and operations are worthy of trust.

In this section, we will present the usual compliance certifications that are used by SaaS Providers to confirm their trustworthiness. We will also discuss other security and compliance topics such as data location & sovereignty, disaster recovery and business continuity planning.

7.1 Three pillars of SaaS GRC

Governance, Risk, Compliance (GRC) is a term used to describe an organization's processes used to address corporate governance, enterprise risk management and corporate compliance with applicable laws and regulations. These three objectives are sometimes referred to as the three pillars of GRC. Some texts will enumerate four pillars, or even twelve pillars; but that is another debate.

Below are some standard definitions for these three pillars which we will then apply to the specific case of SaaS.

Governance: Ensuring that Policies and Strategy are implemented, and that required Processes are correctly followed. Governance includes defining Roles and responsibilities, measuring and reporting, and taking actions to resolve any issues identified. (34)

Risk management: The set of processes through which management identifies, analyzes, and, where necessary, responds appropriately to risks that might adversely affect realization of the organization's business objectives.

Compliance: Management processes which identify the applicable requirements (defined for example in laws, regulations, contracts, strategies and policies), assess the state of compliance, assess the risks and potential costs of non-compliance against the projected expenses to achieve compliance, and hence prioritize, fund and initiate any corrective actions deemed necessary.

7.2 Governance

Governance, as we can conclude from the definition above, is ensuring that work is done correctly, as opposed to doing the actual work. In the startup stage, Job #1 of the Executive Team is to establish the corporate strategy and priorities, then build the team to execute on those objectives. Governance often remains an afterthought until a milestone is reached or a crisis encountered. However, putting in place the proper governance processes can potentially avoid many a crisis during the growth stage.

Governance is centered around the Executive Team. However, the Board of Directors, when one exists, also plays an important role in Governance. The CEO and the Executive Team need to define Strategy, Priorities and Policies, as illustrated in the following figure. They also need to define the Values of the organization; the "tone from the top", which should permeate throughout the organization.

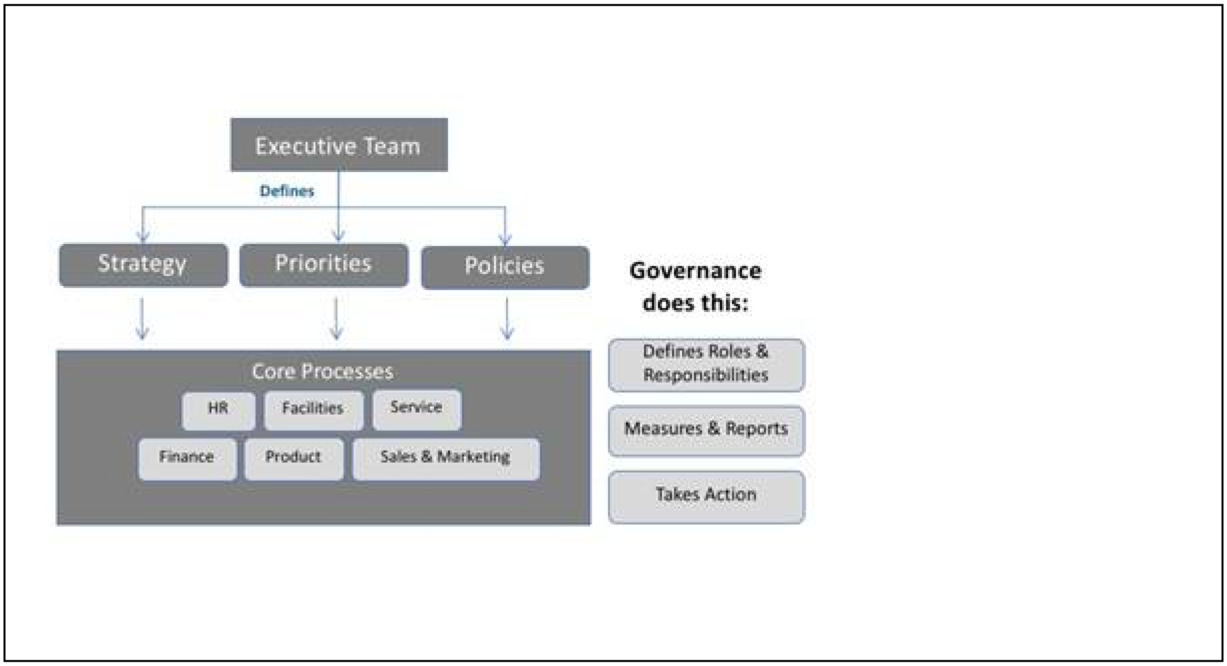


figure 26 - Governance

The CEO and the Executive Team must also find the right balance so that the Governance is adjusted to the needs of the company. In a growth stage SaaS company, the focus will be on new Sales, top-line growth and gaining market share & market recognition. In this case, governance processes need to be more agile and flexible.

In a SaaS company that is at a later stage, the focus will be on standardization, efficiency, effectiveness & scalability, bottom-line growth. The latter stage SaaS company will aim for high customer satisfaction which will lead to higher renewal rates. In this case, governance processes will aim for standardization, cost optimization and predictability of results.

The following figure illustrates five vectors of the governance process that evolve with the growth and maturity of the organization. Each organization is different and will need to determine where to position each slider, to the left or to the right of the scale. There is no "right" answer; each organization must find its own "sweet spot" for a specific context at a particular moment.

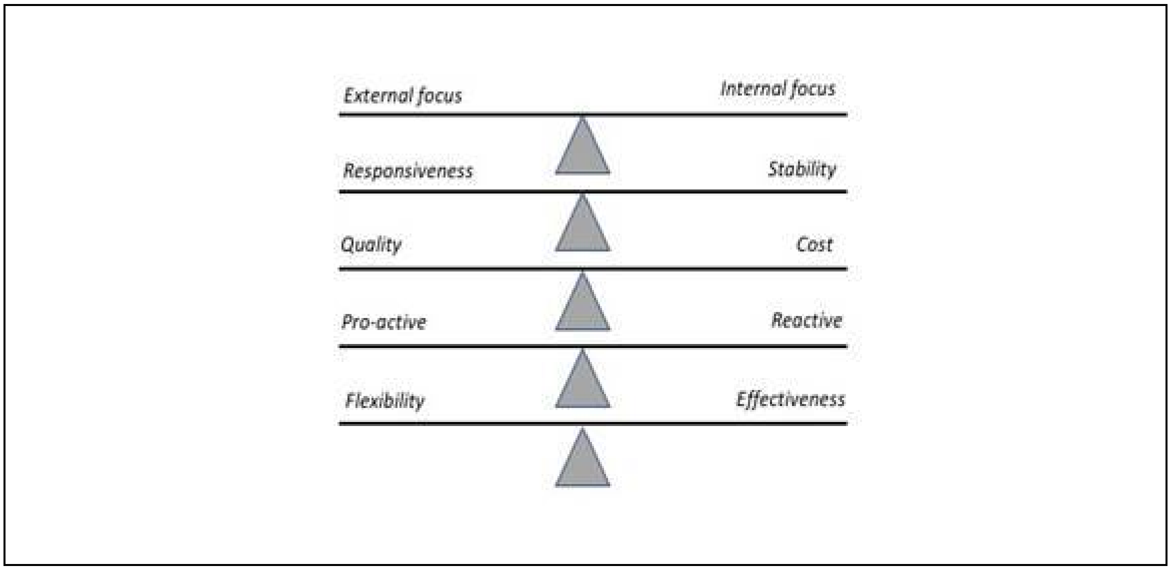


figure 27 - Five vectors

Other elements of the business plan can be thought of as vectors and can influence the five ones above. For example, focusing on top-line growth or on bottom-line growth, focus on acquiring logos or acquiring revenue, etc.

7.3 Risk Management

Risk Management is driven by a set of key policy decisions and the implementation of appropriate Information Security (InfoSec) and operational controls.

Among the key **policy decisions** that need to be made by the SaaS Provider are:

- Disaster Recovery / Business Continuity
- Personnel Policies
- Physical Access Policies
- Data Access / Data Security policies

The key **InfoSec controls** that should be put in place include:

- Vulnerability scans
- Threat analysis
- Access & password controls
- Intrusion detection
- Key management
- Security incident management

The key **Operational controls** that should be put in place to mitigate risk include:

- Log analysis
- Backup validation
- Commissioning/decommissioning processes
- OS hardening
- Security patching

One key characteristic of Risk Management in the SaaS environment is that the SaaS Provider is imputed with the overall responsibility for this, when in fact the SaaS service may rely on the services of other cloud suppliers (IaaS supplier, Colocation provider, CDN, etc.) for certain components of the service.

When selling to Enterprise clients, SaaS Providers should be prepared to answer multiple questions and fill out numerous checklists to comply with their prospective client's security audit requirements. When dealing with sensitive information, that if divulged could put the entire SaaS service at risk, the SaaS Provider should simply state that the information requested cannot be shared for security reasons, or can only be shared verbally, etc.

Some typical questions that might come up during such a client audit are:

- When was your last penetration test, and can we see the results?
- What encryption standards are used and where is encryption utilized: data at rest, data in transit, backups, etc.?
- How are clients notified of security breaches, violations, or suspicious activity?
- What is the policy for applying security patches to operating systems, database systems, middleware, management tools, etc. and is a log of changes kept?
- If the cloud service provider uses subcontractors for any parts of the service, including system administration personnel, do these third parties provide an equally strong level of security?
- If PKI or symmetric keys are used to secure access to the SaaS service, how are the keys managed and protected? How are SSL keys managed and rotated?
- Do you support Single Sign-On (SSO) through a common standard such as OAuth 2.0 or SAML 2.0? A common SaaS Tenant requirement is for integration with their Identity and Access Management (IdAM) system with the SaaS Service. Since it is unpractical for the SaaS Tenant to administer a separate IdAM system for each cloud service that he uses, the best arrangement is for the SaaS Provider to delegate authentication capabilities to the SaaS Tenant's internal IdAM system through an SSO arrangement.

7.4 Compliance

We have defined **compliance** as the management processes which identify the applicable requirements (defined for example in laws, regulations, contracts, strategies and policies), assess the state of compliance, assess the risks and potential costs of non-compliance against the projected expenses to achieve compliance, and hence prioritize, fund and initiate any corrective actions deemed necessary.

There are several standards that exist than can apply to any IT system in general and others that are more specific to cloud-based systems. Some standards are industry-specific and others are aligned towards the type of data that is being processed and stored. The following table gives a partial list.

Standard	Category	Description / Reference
SAE-18	Security	Controls for finance, security, and privacy. Supersedes SAS70. Supersedes SSAE-16 since May 2017 http://ssae-18.org/
irective 5/46/ec	Security	EU Directive on the processing and movement of personal data; will be superseded by the General Data Protection Regulation in May 2018. http://eur-lex.europa.eu/legal-content/EN/TXT/HTML/
irective 002/58/ec	Security	EU E-Privacy Directive http://eur-lex.europa.eu/legal-content/EN/TXT/?qid=1489683968665&uri=CELEX:32002L0058
irective 006/24/ec	Security	EU Data Retention Directive http://eur-lex.europa.eu/LexUriServ/LexUriServ.do?uri=OJ:L:2006:105:0054:0063:EN:PDF
SO 27001	Security	Specification for an information security management system https://www.iso.org/standard/54534.html
SO 27017	Security	Code of practice for information security controls based on ISO/IEC 27002 for cloud services https://www.iso.org/standard/43757.html
SO 27018	Privacy	Code of practice for protection of Personally Identifiable Information (PII) in public clouds acting as PII processors https://www.iso.org/standard/61498.html
OX	Financial	Sarbanes-Oxley Act; US public company financial accountability controls. www.soxlaw.com
CI DSS	Credit Card	Payment Card Industry Data Security Standard https://www.pcisecuritystandards.org
IIPAA	Health	Health Insurance Portability and Accountability Act – standard for the security and privacy of health care data. www.hhs.gov/hipaa/
NSI/TIA-	Data	Security, availability and operational sustainability of

42	Center	data centers. http://www.tia-942.org See also: www.uptimeinstitute.com
edRAMP	Security	Federal Risk and Authorization Management Program; US Government standard for cloud computing www.fedramp.gov
IPS	Software	Federal Information Processing Standard Publication 140-2; US Government Standard for computer systems security and interoperability; and especially data encryption standards. https://www.nist.gov/information-technology-laboratory/fips-general-information
ISMA	Security	Federal Information Security Management Act; US Government standard to strengthen cybersecurity http://csrc.nist.gov/groups/SMA/fisma/overview.html
ERPA	Education	Family Educational Rights and Privacy Act; US Government standard for security and privacy and education information. https://www.law.cornell.edu/uscode/text/20/1232g
ERC	Power Utilities	North American Electrical Reliability Corporation; standards for power utility control systems. www.nerc.com

figure 28 - Sample of GRC Standards

As an example, the following table list the compliance certificates that are maintained by the Google Cloud Platform, an IaaS and PaaS service.

Google Cloud Platform
Certification Footprint (36)

Google has annual audits for the following standards:

SSAE16 / ISAE 3402 Type II:
- SOC 1
- SOC 2
- SOC 3 public audit report

ISO 27001
ISO 27017, Cloud Security
ISO 27018, Cloud Privacy
FedRamp ATO for Google App Engine
PCI DSS v3.1

figure 29 - Google Cloud Platform Certifications

7.5 SSAE 18 Certification

Statement on Standards for Attestation Engagements 18 (SSEA-18)[11] is an auditing statement for service organizations, published by the Auditing Standards Board of the American Institute of Certified Public Accountants (AICPA). This auditing statement replaces the former Statement on Auditing Standards No. 70 (abbreviated as SAS 70) and the more well-known SSAE-16.

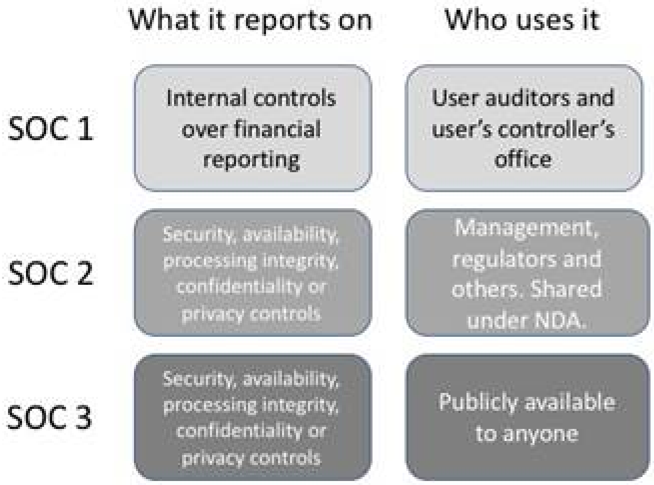
SSAE 18 Certification is based upon a "Service Organization Controls" (SOC) report, of which there are two **types**:

- a SOC Type 1 report is an independent snapshot of the organization's control landscape on a given day;
- a SOC Type 2 report also adds a historical element, showing that controls were managed over time (typically 12 months).

There are three SOC reports, named SOC 1, SOC 2 and SOC 3. A SOC 1 report looks at controls over financial reporting, but a SOC 2 and 3 is an audit of controls at a service organization relevant to security, availability, processing integrity confidentiality, and privacy. For many B2B SaaS solutions, regardless of the industry sector, it is common to see a SOC 1 or SOC 2 as a prerequisite in RFPs. The SOC 2 report is typically the most appropriate for a SaaS solution, but, a SOC 1 is the most requested, although not always the most relevant.

The differences between these three SOC reports are summarized in figure 30 on the next page.

While we still see many technology companies, including some SaaS Providers, performing annual SOC 1 audits this is changing, with more and more opting for the more appropriate SOC 2 compliance.



	What it reports on	Who uses it
SOC 1	Internal controls over financial reporting	User auditors and user's controller's office
SOC 2	Security, availability, processing integrity, confidentiality or privacy controls	Management, regulators and others. Shared under NDA.
SOC 3	Security, availability, processing integrity, confidentiality or privacy controls	Publicly available to anyone

source: (37) (38)

figure 30 - Difference between SOC1, SOC2 and SOC3

We will be focusing on SOC 2, which was specifically designed for entities such as data centers, IT managed service, SaaS Providers and many other technology and cloud-computing based businesses. Within the SOC 2 framework is a comprehensive set of criteria known as the Trust Services Principles (TSP) that are composed of the following five (5) sections:

- The security of a service organization' system.
- The availability of a service organization's system.
- The processing integrity of a service organization's system.
- The confidentiality of the information that the service organization's system processes or maintains for user entities.
- The privacy of personal information that the service organization collects, uses, retains, discloses, and disposes of for user entities. (39)

SSAE 16 provided guidance on an auditing method based on the Trust Services Principles, rather than mandating a specific control set; therefore, it is up to the auditor, often an accounting firm, and the SaaS Provider to determine which reporting framework should be used.

SSAE-18 builds upon the SSAE-16 Trust Services Principles, but places a greater emphasis on the risk assessment process. It also makes the following two components mandatory:

1. Service Organizations will need to implement a formal Third Party Vendor Management Program
2. Service Organizations will need to implement a formal Annual Risk Assessment process

These two elements were typically present in SSAE-16 SOC2 reports, but were not formally required. They now should be mandatory in all SSAE-18 reports.

For the other elements, there are many choices as to what reporting framework to use, such as COBIT (35), COSO (40) and the CSA (41). The Cloud Security Alliance (CSA) is particularly interesting since it targets cloud computing specifically. The CSA Cloud Controls Matrix (CCM) is designed to provide fundamental security principles to guide cloud vendors and to assist prospective cloud customers in assessing the overall security risk of a cloud provider. It contains an excellent set of prescriptive criteria for assessing cloud providers for SOC 2 compliance. The CSA also sponsor a Working Group on SaaS Governance.

Roadmap for SSAE Certification

Achieving SSAE Certification can be quite simple or it can be a long, drawn-out, costly and frustrating process. It all depends on the amount of time and preparation that you are willing to put into the process.

Before calling an auditor to perform a formal assessment, begin with a Scoping and Readiness Evaluation. This step can be done by a SSAE auditor or it can be done by industry specialist who has successfully gone through the compliance exercise. First, determine the scope of your SOC 2 evaluation: which TSPs, which Control Framework, which controls? With that scope in hand, execute a pre-audit evaluation to determine current gaps and deficiencies that will need to be corrected.

Next, perform a pre-audit remediation exercise. There are two parts to this remediation: a **policy and procedure remediation** and a **technical remediation**.

The policy and procedure remediation should entail listing all policies and procedures that will need to be developed to successfully pass the audit. Having all these policies and procedures fully documented is an important pre-requisite to successfully passing the audit. We present a list, at the end of this section, of a minimal set of policies and procedures that should be core to a SaaS Provider. Depending on your context, additional Policies and Processes may be required, beyond this list.

The technical remediation involves eliminating or mitigating any vulnerabilities that may exist in the computing infrastructure. Many of these can discovered by performing a vulnerability scan with a tool such as Nessus, or by hiring a specialized firm to do it for you. Technical remediation would involve things such as upgrading software, installing security patches, reviewing access lists, closing unused ports, reviewing firewall rules, etc.

Sample List of Core Policies & Processes

9 Core Policies to Implement

- Acceptable Use Policy
- Business Continuity Policy
- Change Management Policy
- Data Protection Policy
- Logical Access Control Policy
- Personnel Security Policy
- Physical Security Policy
- Third Party Service Provider Policy
- Vulnerability Management Policy

8 Core Processes to Implement

- Annual Sign off on Acceptable Use
- Business Continuity Process
- Change Management & Control
- Disaster Recovery Process
- Logical Access Control Process

- Policies Review Process
- Policy Exception Request Process
- Vulnerability Management Process

figure 31 – Sample List of Core Policies & Processes

After you have completed this pre-audit remediation, you can then proceed with the formal audit. The number of gaps will have been greatly reduced or eliminated completely, making the audit process much faster and less costly. The cost of an audit can vary greatly, based on the number of controls, size of the company, and complexity of the IT infrastructure.

In conclusion, the primary reason for the SSAE 16 or 18 SOC 2 and SOC 3 audits is to provide assurance to a third party and their auditors that a SaaS Provider is delivering his service with proper controls in place to provide an appropriate level of security, availability, processing integrity, confidentiality, and/or privacy.

7.6 Security Compliance

After SSEA 16 compliance, the second most requested certification will be a security-related certification. The most frequent being ISO 27001 and cloud-specific standards such as ISO 27017 and 27018.

The ISO 27001 is the standard for an Information Security Management System. Like many standards, it is based upon controls and control objectives. In the 2013 version of the standard, there are 114 controls and 35 control objectives. SaaS Providers who are certified ISO 27001 have through a formal compliance audit done by an independent and accredited certification body.

In the same family is the ISO 27002 standard. The ISO 27002 provides "best practice" recommendations on information security management. These recommendations are based on the C-I-A triad of confidentiality, integrity and availability. It can be an extremely useful reference for the establishment of an Information Security Management System. However, it is not a certifiable standard like ISO 27001. Therefore, a SaaS Provider cannot be certified ISO 27002, but can use this standard to implement many best practices in security.

The ISO 27017 standard is a code of practice for information security controls based upon ISO 27002 for cloud services. Whereas ISO 27018 is code of practice for the protection of personally identifiable information (PII) in public clouds acting as PII processors. Both standards are certifiable.

7.7 Disaster recovery strategies

Disaster recovery (DR) is a subject that is often buried when the SaaS Provider doesn't have a great story to tell. How many people have signed up for a free gmail, Hotmail or Yahoo mail account and actually know if there is a disaster recovery clause in the end-user agreement that they signed? What about Disaster Revovery for more "mainstream" services such as Github, Netsuite, or Slack? It might be on page 13 of an online End-user agreement, in font 8 or smaller, and will get little scrutiny in a B2C context.

For a B2B SaaS Service, the question often gets asked more directly, and can cause anxiety attacks for a lot of startups. Mostly startups are cash-constrained and time-constrained as well, as they rush to integrate new features, onboard new clients and scale out their infrastructure. Little wonder that no one has the time or money to think about disaster recovery.

The reality is that many clients will ask about and demand world-class disaster recovery, but very few will be willing to spend extra money for that capability. However, in the life of most SaaS companies, especially in the B2B space, the question of DR will inevitably come up sooner or later. As the business and revenues scale, a prolonged outage due to a disastrous event may put the entire business at financial risk. That is when the Board or the Executive Management Team will insist on addressing the problem.

Disaster Recovery Metrics

In designing a disaster recovery solution, the SaaS Provider needs to identify three key metrics: RTO, RPO and the cost of downtime.

Recovery Time Objective (RTO) is the maximum period within which service must be restored after a disaster. For a business-critical function, like a user authentication service or a credit card processing application, an acceptable RTO could be minutes. In other cases, where no revenue or reputation loss is in danger, RTO could be days or weeks. SaaS Tenants should be aware that the SaaS Provider will often calculate the RTO beginning at the point in time where a disaster has been declared and the DR process has been initiated. This may be several minutes or several hours after the beginning of the actual service interruption.

Recovery Point Objective (RPO) is the maximum period for which data may be lost due to a disaster.

For example, in a system where recovery backups are taken nightly, the RPO would be 24 hours. Where there exists a paper trail of all transactions made during the day, the SaaS Tenant may be able to re-key all the “lost” transactions resulting from a disaster. In such a case, a RPO of 24 hours would be inconvenient but with moderate business impact.

For other business applications, like our credit card processing example, RPO tolerance might be very low or nil.

The **cost of downtime** is another important variable since it will confirm the cost-justification of putting in place the DR strategy. The downtime cost can be from several sources, such as lost revenue, cost of falling back to a manual process or reputational cost.

Disaster Recovery Strategies. There are several DR strategies for SaaS. The choice will be dictated in a large part by the RTO/RPO objectives. The foundation for any DR strategy is access to the data, configuration files and programs of the service. Therefore, a remote backup strategy underlies each DR strategy. A SaaS provider should also implement an on-site backup strategy, which may prove useless in the case of a disaster, but is essential to answering certain types of data restoration or roll-back requests.

The classic **remote backup and restore** strategy is least costly of all the strategies. At the primary site, full and incremental backups are taken. These are then copied to a secondary site. If the primary site is struck by a disaster, the service can be restored at the secondary site by rebuilding the infrastructure stack and reloading the data. Since there are no servers running at the secondary site, costs are less. However, the RTO will be extremely high because of procurement and installation delays. RTO will be less if the infrastructure can be built-out dynamically using an IaaS service, however, the time required to restore databases from backups can still take hours or days depending on the database size.

An **active-passive** solution will have the servers and database servers already provisioned at the secondary site. The secondary site can be **cold** or **warm**. A cold site means that the servers are not powered on. When a disaster is declared, the IT Ops group will run a series of scripts to power on and configure the infrastructure. Then the databases are loaded from the latest backups and service can be restored. Active-passive DR with a cold site is a good solution when RTO is not too stringent. It can also be very cost-effective if there is equipment at the secondary site that can be repurposed; for example, development, test or staging servers.

A warm site means that the servers are running and that the database is kept in sync with the Production server. This strategy is the one to use when RTO and RPO requirements are very short. There are several technologies available to keep two databases in sync (for example, Rsync, DataGuard, GoldenGate, etc.). They usually function with an initial full backup, then transaction logs from the primary server are transmitted to the secondary site and applied to the standby database. The interval at which the transaction logs are shipped can be parameterized to meet the RPO objective. Thus, the maximum data loss will be the interval between log shipments, which can be a matter of minutes.

The final strategy that we would like to discuss is the **active-active** scenario. In this scenario, the two data centers are both active and both are processing transactions in production. They mutually assure a fail-over capacity, one for the other. Data Center A will the primary data center for a group of tenants, and is also the backup datacenter for those tenants hosted in Data Center B. This scenario is the costliest since each tenant utilizes infrastructure capacity in the primary and the backup data center. Each transaction is also processed twice, once in the primary data center, and once again in the secondary data center, to keep the secondary data center in sync. However, with this scenario it is possible to deliver extremely short RPO and RTO times.

With the two first strategies, backup & restore and active-passive, the SaaS Provider should perform a DR test on a regular basis (typically once or twice a year) and be willing to share the results of those tests with those SaaS Tenants that request it. The DR test, at a minimum, demonstrate that Tenant data can be restored at the DR site and that the client service can be restored with the same configuration as the production site.

Tenants should not expect the SaaS Provider to do a full cut-over and actually run his production services from the DR site for a period of time. The RPO and RTO objectives might be acceptable in the case of real disaster, but unacceptable when compared to normal production service levels. Secondly, doing the roll-back to the original production site, after a complete cutover, involves a complete re-synchronization of the data bases between the two sites. This operation can take hours or even days depending on the size of the databases. Third, the DR site will be reachable through a different IP address than the production site. This has two unavoidable consequences:

- the SaaS Provider will need to do a DNS update to associate the URL of the service, i.e. www.saas-service.com, to the IP address of the DR site. This DNS update will take a certain amount to propagate through the Internet;
- any SaaS Tenants who only allow white-listed IP addresses through their firewalls, will need to ensure that the DR site IP address range is added to their firewall rules.

Regardless of which DR scenario the SaaS Provider adopts, all SaaS Tenants should also do their own disaster recovery planning. Each business has unique requirements and the disaster recovery plan needs to be tailored to their specific business objectives.

7.8 Data residency and data sovereignty

The SaaS provider should choose his hosting locations carefully. As discussed in Chapter 5, the geographic distance between the hosting center and the user community can introduce response time latency.

However, another important consideration in choosing a hosting location is data sovereignty. In an ideal world, every client would like to have his data reside in his own country. Building out 200+ hosting centers is not a cost-effective strategy, so what factors should be considered when deciding on a hosting location?

When determining data residency, not only should the data storage location be considered, but also data in transit, location of backups and location of disaster recovery sites. The impact of these additional locations might be mitigated if all data is encrypted, at rest and in transit.

Two important considerations for data sovereignty are the U.S. Patriot Act and the EU Data Protection Directive.

7.8.1 Patriot Act

The U.S. Patriot Act was enacted shortly after the attacks on the World Trade Center in 2001. Its objective is to enhance the ability domestic security services to prevent terrorism. It contains multiple measures: anti-money laundering, border security, terrorism criminal law, etc. However, for a SaaS Provider, the important provision is the permission given to authorities to compel organizations to provide access to data that they possess.

Although the Press often describes the Patriot Act as a sort of magic wand that gives unfettered access to all data, it is important to understand that the Patriot Act is in fact a series of amendments to existing laws; in many cases, it streamlines the process to obtaining information, but while respecting the underlying laws and regulations pertaining the government requests to access data that were in effect before the Patriot Act.

Shortly after taking office, President Trump signed an executive order, ***Enhancing Public Safety in the Interior of the United States***, that effectively removed any privacy protections for foreigners. It is still too early to measure the effect that this will have on Internet communications and SaaS services.

The Patriot Act anxiety is particularly great in non-U.S. countries: specifically Canada (42) and Europe. However, avoiding the reach of the Patriot Act is not easy. If an EU company stores its data on the servers of an American-based cloud provider, its data will be subject to government access through the Patriot Act. If the same EU company stores it data on the servers in Europe of a company that also has operations in the U.S., may also be subject to Patriot Act discovery tools.

If an EU company with no US presence hosts its data with an EU cloud provider that also has no US-presence, then it is possible that the data may be beyond the direct reach of the Patriot Act. In reality, the data may still be accessible to the U.S. government through a mutual assistance treaty or a national security information sharing treaty, that are in existence in many countries.

Note that even if your data isn't stored in a US cloud service, if it is transferred online through the US, it may be collected by the US government. It is estimated that 90% of Canadian Internet traffic is routed through the US. So, a Canadian SaaS Tenant may unknowingly be exposing his data, even if he has carefully chosen a SaaS Provider with a Canadian hosting location.[12]

Various privacy legislation in Canada that applies to private companies has stipulations requiring that companies receive consent from the owners of personal data before they store or transmit it outside of Canada's borders, for example British Columbia's Freedom of Information and Protection of Privacy Act. It requires data remain in Canada in transport unless unambiguous consent is provided in an end-user contract. However, having no control over the routes chosen by the major carriers, it is hard offer up guarantees that data will be transported completely within Canadian borders.

As we will see in the next chapter, the SaaS Contract should include a clause stating how the SaaS Provider is required to respond to government requests for data, including those requests under the Patriot Act.

7.8.2 Safe Harbour

If your SaaS Service stores personal information and your SaaS Tenants have end-users in the European Union (EU), you should be aware of EU Data privacy laws and the Safe Harbour rules.

The EU has developed a very strict set of rules for the protection of the personal data of EU citizens. According to the **Data Protection Directive**, companies operating in the European Union are not permitted to send personal data to "third countries", outside the European Economic Area, unless they guarantee adequate levels of protection or "the data subject himself agrees to the transfer". Normally, to guarantee adequate levels of protection, the European Commission must declare the full "adequacy" of a country's data protection laws.

The Safe Harbour Privacy Principles were developed between 1998 and 2000 to answer this question of “adequacy”. US companies could opt into the program and be certified if they adhered to seven principles and 15 frequently asked questions and answers per the Directive. In July 2000, the European Commission (EC) decided that US companies complying with the principles and self-certifying that they met the EU requirements, the so-called "safe harbour scheme", were allowed to transfer data from the EU to the U.S. This is referred to as the **Safe Harbour Decision**.

However, an EU Facebook user complained that his personal data was insufficiently protected. In October 2015, The European Court of Justice declared that the Safe Harbour Decision was invalid which lead to talks between the U.S. and the EU to find a new framework for transatlantic data flows.

This new framework is called the EU-US Privacy Shield and went into effect in July 2016. U.S. businesses can self-certify through the ww.privacyshield.gov website.

If a SaaS Provider does not wish to self-certify under the Privacy Shield rules, then the only other option is likely to be opening a data center in Europe (or use a European instance of a IaaS service) and store all EU citizen-related data there (and ensure that all backups remain in Europe and the DR site is also in Europe).

8. YOUR SAAS CONTRACT

I have no contracts with my clients; just a handshake is enough.
- Irving Paul Lazar (talent agent & dealmaker)

There is no uniform model for a SaaS contract. There are, however, several key subjects that are generally recurring themes in just about all SaaS contracts. The complexity of the SaaS contract will depend on several criteria, such as the type of service (enterprise or consumer), the sophistication and experience of the SaaS provider, the sophistication and experience of the SaaS subscriber, the business criticality of service and the sensitivity of the data.

In this section, we will discuss the habitual components of a SaaS contract, the key variations and their impact[13]. SaaS contracts are generally drafted with the intent to specify what services will be provided and to protect the SaaS provider from litigation. Thus, we will describe the zones of risk that a SaaS provider will want to mitigate in his contract.

Likewise, we will also discuss the point of view of the SaaS tenant. There are many clauses of a SaaS contract that the SaaS tenant will want to review, and negotiate if possible. The SaaS Provider will leave out anything that is potentially unfavorable to them; the SaaS Tenant should be aware of those possible, often intentional, omissions. In many cases, the contract may be non-negotiable, but it is still important to understand the implications of each clause. Obviously, larger SaaS Tenants may have additional bargaining power and in some cases, the contract changes that they negotiate may trickle done to the contracts of smaller clients.

The SaaS Provider should aim to write an agreement that is complete, from a legal point of view, but that is not intimidating, so that a SaaS Tenant can read it and understand it without having an army of lawyers by his side. Service Agreements written in plain language and a friendly tone are much more likely to be read, and understood, by your SaaS Tenants.

8.1 Services Agreements

A SaaS services agreement is important since it establishes the baseline of expectations and, more importantly, the obligations of both the SaaS provider and the SaaS tenant. The SaaS services agreement may be structured into several separate documents such as: Master Services Agreement, Service Level Agreement (SLA), Acceptable Use Policy (AUP) and a Privacy Policy. The Master Services Agreement can be presented by several different names, such as: Terms of Service, Agreement, Terms & Conditions, User Agreement, etc.

In many cases of a Business to Consumer (B2C) service, the services agreement will be placed on a web site with an AGREE button underneath. In this case, the tenant does not have the opportunity to negotiate the terms of the agreement. Nonetheless, such an agreement should be read and understood.

In a Business to Business (B2B) service, the services agreement may or may not be negotiable, depending on the importance of the contract and the parties to that agreement. A large client will have more bargaining power during the contract signing than a small client. Likewise, a small SaaS provider will usually display more flexibility than a larger, more established, provider. However, the small provider may over-promise service levels are beyond his capabilities.

In both cases, negotiable or non-negotiable, the SaaS provider needs to draft a services agreement that protects him and defines clearly what services he is obligated to provide. Likewise, the SaaS tenant needs to understand the obligations of the SaaS provider and what services/guarantees are not part of the contract.

The baseline content of the Services Agreement should touch on the following subjects:

- **Service Definition**, including usage limits or number of authorized users[14];

- **Service Levels**, which defines thresholds for availability, serviceability or performance. This section might also specify penalties if those thresholds are not met;
- **Fees and Payment**; which describes the methods for paying for the service;
- **Data Management**, including topics such as encryption, data residency, access controls, etc.
- **Support**, who do you contact with questions or issues, how fast will they respond and do you get to talk to a real human being or only a chatbot ?
- **Indemnification**, hopefully you never have to use these clauses unless something really bad happens,
- **Termination**, like a pre-nuptial arrangement, you need to plan for the day when you decide to part ways.

8.1.1 Service Definition

The Service Definition will describe which services are made available to the SaaS tenant. This definition can be especially critical when the SaaS service is comprised of several independent or optional modules.

The service definition may include the number of authorized users and other such usage limits. Alternatively, these quantities may be specified in a transaction document or "order form" attached to the Services Agreement.

Who is an "authorized user" ?

Here is an extract from GitHub's Terms of Service:

" You must be a human to create an account. Accounts registered by "bots" or other automated methods are not permitted. We do permit machine accounts.

Your login may only be used by one person — i.e., a single login may not be shared by multiple people. A paid organization account may create separate logins for as many users as its subscription allows.

Overall, the number of Users must not exceed the number of accounts you've ordered from us.”

So obviously, GitHub had experienced some challenges in the past with “bots” automatically creating accounts. Make certain your users are human, especially if you are offering free, trial or freemium resources!

The pricing structure should be simple, easy to understand and easy to monitor. Let’s illustrate where it may get complicated.

The SaaS Provider will seek to match pricing with his costs of providing the service; this often leads to introducing elements of variable pricing. Even though elasticity is a fundamental characteristic of SaaS, the SaaS Provider will also want to protect himself from excess usage by imposing thresholds or incremental rates.

For the SaaS Tenant, variable rates introduce a level of uncertainty in billing. To avoid surprises, the SaaS Tenant should request that the SaaS Provider make available tools to monitor usage and provide auditable justification for variable costs. If the SaaS Tenant is unable to control usage then the SaaS Tenant should negotiate a usage cap or a billing cap to avoid costly accidental overruns.

SaaS Tenants should take the time to properly estimate their usage and size their contract accordingly. They should understand what happens if they exceed their usage thresholds and have the ability to renegotiate their usage levels during the life of the contract. **Usage optimization** describes the rights SaaS Tenants should expect from their SaaS Provider as their business changes; how they expand, maintain or contract in their usage of the solution.

SaaS Tenants should also be able to provide access to the SaaS service to majority-owned affiliates under the same terms as their own internal users.

8.1.2 Service Levels

This section should describe the SaaS Provider's commitment to provide for **availability**, **serviceability** or **performance**. Many SaaS Provider's contracts will contain very vague language regarding these questions. It is up to the SaaS Tenant to obtain the necessary clarifications.

Obviously, the type of service will influence the service levels offered by the SaaS Provider. If the service is a free trial or a Freemium service, it will probably come with no service level commitments. However, if the service being delivered is mission critical, for example an authentication service, then the service level commitments will be very high.

Availability is usually indicated by an uptime percentage, for example 99.99% availability. It is important to understand how that number is calculated, what is included and what is excluded from the calculation. 99.99% availability measured daily is very different from an average calculated over a month or a quarterly basis.

The method of measurement is important. A ping or a POST to the login page of the service is easy to do, but only gives a partial vision of availability. A successful ping will not detect an outage due to a faulty App Server or Database Server. A synthetic transaction that traverses all the system components up to doing a READ on the database is the only way to ensure that the system is fully operational.

Also, the test interval is also important to know. Is downtime counted after one unsuccessful ping, or only after 2 unsuccessful pings? If we ping the service every 10 minutes, we will not detect outages, up to nine minutes long, that are completely between two pings. An unsuccessful ping detects an outage, but we will not know the exact duration since the outage may have started up to nine minutes previously. The measurement methodology can have a big impact on the accuracy of SLA results reported.

Understanding how availability is measured is also important when the SaaS offering is comprised of different sub-services. If the main application is functioning correctly, but an email generation component or a reporting component is defective, is it counted as an outage?

Some SaaS providers will offer service credits if the SLA targets are missed. Some important points to note:

- In most cases, the SaaS Provider will offer a credit on future invoices and will not provide a refund. Therefore, if the SaaS tenant decides to leave a SaaS provider because of poor performance, the Service Level credits will be of no use;
- Usage credits are usually capped. For example, at 50% or 100% of one month's billing;
- Who is responsible for monitoring SLA levels? In some cases, the burden of SLA violation notification rests with the SaaS tenant who may have to put in place some form of monitoring and have a process for requesting SLA credits.

The SaaS Provider should provide transparency on the availability results and should make these statistics available to its Tenants. However, it is still common to see Service Agreements with no commitment as to availability and where the burden of proof of outages rests with the SaaS Tenant.

Serviceability refers to the practices for installing upgrades, patched and fixes. It is up to the SaaS Tenant to judge whether the maintenance/upgrade policy of the SaaS Provider will meet his business requirements.

Is there a dedicated window that is reserved for maintenance/upgrades? Is there downtime involved in performing maintenances and upgrades? In most cases, any period reserved for maintenance activity will be excluded from the SLA calculation. Too frequent and too long maintenance periods may be disruptive to business operations. For a SaaS Tenant with global operations, downtime associated with maintenance operations can be problematic: a maintenance window of 2 am to 3 am in Silicon Valley corresponds the start of the business day in Europe.

Are upgrades automatically pushed into production? or does the SaaS tenant have the option to upgrade when he wants? For certain industries who traditionally have blackout periods (like the Holiday shopping season for the retail sector) this can be a big departure from their usual practices.

Other exclusions can be emergency maintenances or security patches announced x hours in advance. Although it is to everyone’s advantage that emergency patching gets done on a timely basis, the prudent SaaS customer should validate that the SaaS provider does not make an abusive use of such a clause.

SLAs around **performance** usually indicate limits around response times or latency. For these clauses to be meaningful, the SaaS tenant needs to confirm that these measures correspond to his business requirements and that the way that the metrics are measured is an adequate representation of the user experience.

8.1.3 Fees and Payment

The contract should specify how and when payment for the service is to be made. Monthly billing on a credit card is typical, but some enterprises will prefer to receive an annual invoice with payment terms of 30, 45, 60 days.

Some services will invoice the upcoming month in advance. However, when there is a variable component to the pricing model, invoicing for the past month becomes the standard model.

There may be additional language concerning late payments and if the service can be suspended for delinquent accounts. Additionally, the process to apply service credits for outages should be described.

Tax laws can be complicated. It is wise to include a paragraph stating that you will collect any taxes that you are obligated to, without specifying exactly which taxes. The taxes that you need to collect will depend on the country in which you have registered your SaaS company and the country in which your SaaS subscriber is located. And even this can change over time. Set aside some money in your budget to consult a good tax lawyer.

8.1.4 Data Management

This is an especially important topic in a SaaS contract. There are several aspects to this question that we will discuss below.

Data Ownership

The contract should protect the SaaS tenant by explicitly stating that the SaaS tenant retains ownership of his data. The SaaS provider is merely a custodian of the data. Therefore, the SaaS Tenant retains all rights, title and interest in their content and data, and their end-user's use of that content and data.

Similarly, the SaaS Provider should protect himself by excluding types of data that his application was not designed to manage. For example, if the application is not HIPAA-certified, then the SaaS Provider should indicate in the contract that the tenant should not upload this type of data. The SaaS Tenant remains responsible for any personally identifiable information (PII) that they store in the service and they should agree to comply with all applicable privacy and data protection laws.

Customers should ensure that their data will be available if the SaaS Supplier declares bankruptcy protection. Similarly, if the SaaS Provider suspends access to its service because of a billing issue or security issue, the customer’s data should not be held hostage while the dispute is being resolved.

Data Privacy & Access

The service agreement should explicitly state that the SaaS Provider will not access the customer’s data. There may be exceptions to this, for example for database management, but these should be listed explicitly (whom?, under what circumstances?, are accesses logged?, are personal screened?, etc.).

The service agreement should also state whether data is encrypted, at rest and in transit, and which encryption technologies are used.

Data Residency

Data residency is defined by the Object Management Group as “the issues and practices related to the location of data, movement of data across geographies and jurisdictions, and protection of that data against unintended access” (43). This is a somewhat broader definition that just “data location”.

It is legitimate that the SaaS Tenant will want to know where their data will reside and whether this location is fixed or can change at the SaaS Provider’s discretion. However, it is the SaaS Tenant’s responsibility to appreciate the sensitivity of their data to its location and ask the appropriate questions of their SaaS Provider.

In many countries throughout the world, numerous laws, regulations, and other mandates require public and private organizations to protect the privacy of personal data stored in computer systems.

When data is transferred to a cloud, the responsibility for protecting and securing the data typically remains with the controller or custodian of that data, even if in some circumstances this responsibility may be shared with others. The controller of the data may remain liable for any loss, damage, or misuse of the data, even if it relies on a third party to host or process its data.

For example, is the SaaS Tenant storing personal information about European Union citizens? In that case, the SaaS Provider needs to understand the data protection requirements of such data and should be able to describe the data residency requirements of all countries where the data might end up residing. This analysis should include all locations where the data could potentially reside. For example, copies used for backup and disaster recovery, copies used for testing or reproducing problems by Support,

Therefore, the SaaS contract should contain clear language about where data will reside, or could potentially reside. If the SaaS Provider has infrastructures in multiple locations, the SaaS Tenant should have the option to specify in which locations they wish their data to reside, or not reside.

Data Seizure

Some jurisdictions allow law enforcement and other agencies to seize data under certain circumstances. Therefore, the SaaS contract should include an exception that allows the SaaS Provider to access the SaaS Tenant's data if it is required to comply with such a legal request. If this happens, the SaaS Provider should have the obligation to inform the SaaS Tenant of such an event. A timely notification would allow the SaaS Tenant to seek a restraining order or at least know that the data was accessed and notify their clients if required.

In some cases, however, the Data Seizure mandate may come with a "gag order" that prevents the SaaS Provider from notifying his tenants, or anyone else, that a data seizure occurred. For example, the

California Electronic Communications Privacy Act mandates that in certain cases concerning electronic search warrants that "[...] The court shall issue the order [prohibiting any party providing information from notifying any other party that information has been sought]"

8.1.5 Disclaimer, Indemnification, Limitations

These sections are often non-negotiable and skewed strongly in favor of the SaaS Provider who wants to protect himself against various claims, damages, losses and litigations.

A **Disclaimer** section will describe what is **not** included in the agreement. It might also appear as “Warranties and Disclaimer”.

Indemnification identifies the situations where the SaaS Provider will defend the SaaS Tenant from certain claims, and likewise the situation where the SaaS Tenant is required to defend the SaaS Provider from certain claims.

Limitation of Liability specifies the limits on the compensation amount that can be claimed for certain events or breaches.

Let us look more closely at these three clauses.

The **Disclaimer** clauses will often state that the service is offered “as is” with no other guarantees implied. The following example comes from Netsuite’s Terms of service.

> 4. Disclaimer of Warranties. EXCEPT AS STATED IN SECTION 3.1 AND 3.2 ABOVE, ORACLE DOES NOT REPRESENT THAT CUSTOMER’S USE OF THE SERVICE WILL BE SECURE, TIMELY, UNINTERRUPTED OR ERROR-FREE OR THAT THE SERVICE WILL MEET CUSTOMER’S REQUIREMENTS OR THAT ALL ERRORS IN THE SERVICE AND/OR DOCUMENTATION WILL BE CORRECTED OR THAT THE

> OVERALL SYSTEM THAT MAKES THE SERVICE AVAILABLE…. (44)

You can find a similar example from Google, in section 8.5.

The intent of the **Indemnification** clause is to hold each party harmless from any claims arising from a third-party claim. Typically, the SaaS Tenant must defend the SaaS Provider from any third-party claim that the customer data infringes a copyright or trademark, misappropriates any trade secrets or infringes any law.

When the Indemnification clause is reciprocal (and the SaaS Tenant should verify that it is) the SaaS Provider will defend the SaaS Tenant from any similar third-party claims.

The **Limitations of Liabilities** clauses usually protect the SaaS Provider and limit the compensation that a SaaS Tenant can claim in the event of a breach of contract. This section usually excludes any liability arising from deletion, damage or destruction of the SaaS Tenant's data or any loss caused by the inability to access and use the service.

The clause will set a maximum liability and/or a total aggregate liability. This is usually set as a multiple (typically between one and five times) of the fees that the SaaS Tenant has paid in the last 12 months.

To better appreciate this clause, it is important to know the governing law of the agreement, usually stated in a separate clause. Indeed, most jurisdictions do not recognize limitations of liabilities when the SaaS Provider has been grossly negligent. Several jurisdictions also have statutes preventing unreasonable limitations.

8.1.6 Termination and renewals

It is important to realize that, even though the SaaS tenant has selected the SaaS service as the one that best meets his business needs today, there will come a time when it might be best to terminate the relationship; whether it is because of changing requirements, costs, or dissatisfaction with the service. Therefore, the SaaS tenant should envision what will be required to terminate the service gracefully and possibly migrate to a new service with a minimum of business disruption.

Most often the termination clause will refer to two distinct situations: Termination for Cause and Termination for Convenience.

Termination for Convenience refers to a conscious choice to end the service. In month-to-month contract, the SaaS tenant would be able to terminate the service by giving thirty days' notice to the SaaS Provider. In a one-year or a multiyear contract, the termination for convenience options for the SaaS tenant may be more limited or non-existent, or there may be a penalty for early termination.

SaaS tenants should know that Termination for Convenience clauses can be reciprocal. That is, is the SaaS tenant can terminate the service with thirty days' notice, the SaaS Provider would have a similar right to terminate with no liability to the SaaS tenant.

Termination for Cause allows the SaaS tenant or the SaaS Provider to terminate the contract if there is a material default or breach of the agreement, and that the faulty party failed to correct the breach with a reasonable amount of time (typically 30 days) after being notified. The contract may specify which means of notification (letter, email, etc.) are appropriate.

This Clause is especially important for the SaaS Provider to protect himself and his service from a "rogue client". In certain cases, like a security breach, the SaaS Provider may have the right to suspend the service immediately to protect itself or other SaaS tenants.

Some additional points to consider:

Explicit Term. As with any contract, the term of the contract should be explicitly stated. Does the contract automatically renew at the end of the term? Does the SaaS tenant need to provide notice in writing, 60 days or 90 days before the end of the term, to avoid the automatic renewal?

SaaS Tenant Data. A critical aspect in a SaaS contract is what happens to the tenant's data upon contract termination. In theory, the SaaS Provider should return the data to the SaaS tenant at the end of the contract. The contract should specify whether this is done automatically or only upon request; is there a cost to prepare the data extract? in what form will the data extract be delivered? in a flat file, a CSV file, a database dump? Will the SaaS Provider assist with the migration process?

Once the data dump has been delivered to the client, there is typically a set retention period where the SaaS provider retains a copy of the SaaS tenant's data. This period allows the SaaS tenant to confirm that the data obtained is readable and complete. After the retention period, what is the process for deleting the client's data from the SaaS Provider's storage media? Can a certificate of destruction be obtained that confirms that all SaaS Tenant data has been destroyed?

Business Failure. No SaaS Provider likes to highlight the possibility of going out of business and ceasing operations. Therefore, it is unlikely that such a clause will be in the SaaS Provider's standard contract. However, the SaaS Tenant should consider what would happen if such an event did occur and how he would recover the service and his data.

Transferability. Mergers and acquisitions do happen. SaaS Tenants should consider what rights they should have laid out in the contract to facilitate these transitions.

For example, if the SaaS Tenant acquires a company who is also a user of the SaaS Provider's service, can he combine the two contracts and obtain a larger volume discount? If the SaaS Tenant sells his business, can the contract be transferred in its entirety to the acquirer? If the SaaS Provider is acquired and the SaaS Tenant has a valid business reason for not wanting to do business with the acquiring company, can he terminate the contract and migrate to a new service without penalty?

Renewals. SaaS Tenants should pay close attention to the renewal clause in their Service Agreement. They should understand under which conditions the service is renewed. Many Service Agreements have an automatic renewal clause that is executory unless there is a 30-60-90 day cancellation notice sent by the SaaS Tenant. In some case, the renewal is done under the same terms as the existing contract; in some cases, the SaaS Provider can increase rates by a few percentage points.

SaaS Tenants often overlook their obligation to send a cancellation notice until it is too late. Renewals can be a great opportunity to renegotiate rates or service levels, or even switch providers. By letting their Service Agreement renew automatically, the SaaS Tenant foregoes this opportunity.

Cost of switching SaaS Providers

For the SaaS Tenant, the cost of switching provider remains ambiguous at best and expensive at worst. While the SaaS Tenant has access to and ownership of his data, the hurdle in moving from one SaaS Provider to another increases with usage over time. Without rights over the application's functionality, SaaS Tenants face lock-in if they cannot easily export their business processes that are instantiated in the SaaS Provider's functionality. Add in different architectural standards, varying granularity of process flows and complex metadata models, and SaaS Tenants face an expensive and daunting challenge switching from one vendor to another. Most migration plans are unclear on how to successfully switch from one vendor to another, and are left to be discovered when the need arises.

8.1.7 Support

This section of the SaaS Contract should describe the mechanisms that the SaaS Provider makes available to assist the SaaS tenant with any usability issues. For example, support may only be offered via email, or the SaaS Provider may make a toll-free 1-800 number available for Support questions. Online chat is another option.

The SaaS provider may specify different service levels based upon the severity of the issue. For example, a Severity 1 issue would be raised when the system is totally unusable for all users. A Severity 3 might be only a single user being unable to work, while a Severity 5 would be simply a suggestion for a functional improvement.

The support service levels will typically specify two SLAs, one for the time to respond and another for the time to resolve or restore service. A response only means that someone from the SaaS Provider has acknowledged your service request, for example, via an email response or a phone call return. Whereas resolution means that the SaaS provider has responded with a fix or a work-around which allows the SaaS Tenant to perform his work normally.

Another important point in this section is the availability of the Support Service; the ideal being 24x7x365, meaning 24 hours per day for 7 days a week for 365 days a year, especially for global organizations. Other common variations are 8x5 (business hours on week days) or 11x5 (extended business hours).

Terms like “regular business hours”, “holidays”, “weekends” can all hide surprises. For example, If the Support organization is located on the west coast of the Unites States and operates during regular business hours, then a user community in Europe will be underserved. Similarly, holidays vary from country to country: Canada celebrates Thanksgiving in October while Americans celebrate this holiday in

November. Even weekends can take on different meaning; Friday and Saturday being the normal weekend in several Muslim countries.

For an Enterprise service, the Services Agreement should specify the contact information of the key stakeholders at both the SaaS Provider and the SaaS tenant. If the normal support process is ineffective, then the escalation process should be described.

Change Management. This section will often contain a description of the process used for Change Management and the notification of changes. The Services Agreement should require a minimum notification period in advance of the change. In many cases, the SaaS Provider will “push” updates to his software to keep his client base on the “current version” and not have to support older versions of his software. In some cases, the SaaS Tenant may have the option to “opt-out”, to delay the upgrade, or to have any new features “turned off” initially. However, the SaaS Provider will reserve the right to push an urgent security patch, with little or no advance notice and with no opting-out clause.

8.1.8 Service Agreement Governance

As the SaaS Provider makes changes and improvements to his service, there may be a requirement to modify the Service Agreement. There should be a formal mechanism to inform clients of these changes. For example:

- When the Service Agreement is online with an “I Agree” button, then the standard process would be to “pop up” the next Service Agreement on the next login attempt and force an new “**I Agree**” acceptance before continuing;
- In the case of a written contract, clauses that may change during the life of the contract will usually be contained in a separate document that is referenced to in the contract.

Here is an example from the Netsuite Terms of Service:

> **8.2 To Applicable Terms**. If Oracle makes a material change to any applicable URL Terms, then Oracle will notify Customer by either sending an email to the notification email address or posting a notice to the administrator in Customer’s account. If the change has a material adverse impact on Customer and Customer does not agree to the change, Customer must so notify Oracle via legalnotices@netsuite.com within thirty days after receiving notice of the change. If Customer notifies Oracle as required, then Customer will remain governed by the URL Terms in effect immediately prior to the change until the end of the then current subscription term for the affected service(s). If the affected service(s) is renewed, it will be renewed under Oracle's then current URL Terms. (44)

In this example, “URL Terms” are defined as “the terms with which Customer must comply, which are located at a URL, referenced in this Agreement and are hereby incorporated by reference.”

8.2 Acceptable Use Policy

An Acceptable Use Policy (AUP) is a commonplace contractual element. The AUP should describe clearly how the SaaS Tenant may use the service and will spell out those activities which the SaaS Provider considers to be an abusive or even illegal use of their service. This policy is usually non-negotiable and skewed in favor of the SaaS Provider.

The AUP should describe what actions the SaaS Provider may take in the case of a breach. The SaaS Tenant should take ascertain that this process does not allow the SaaS Provider to terminate the service arbitrarily.

A typical process is that the SaaS Provider informs the SaaS Tenant, in writing or email, of the nature of the breach and allows the SaaS Tenant a certain period of time to stop or correct the behavior. The reasons could be abnormal use of the service, a security risk, or delinquency in payments. After expiry of the grace period, the SaaS Provider will have the right to impose a **Temporary Suspension** of the Service.

In the case of a severe security breach, the SaaS Provider may reserve the right to suspend services immediately. Although it is legitimate for the SaaS Provider to want to protect himself, there should also be an escalation mechanism to resolve the issue quickly if the SaaS Tenant was acting in good faith or was not actually at fault.

The SaaS Tenant should also understand the implication of this AUP if they are granting access to the service to their own clients, over whom they have less control.

8.3 “Force Majeure” and other exclusions

A Force Majeure clause is a standard clause in many agreements that most people consider to be ‘boiler plate’ and, therefore, they don’t spend a lot of time thinking about it’s content. As a SaaS provider, the Force Majeure Clause should protect your business from costly claims resulting from catastrophic events that are totally out of your control.

A Force Majeure clause relieves a party of performing its contractual obligations if an event occurs that could not have been reasonably foreseen by the party affected and is beyond the party’s control. Sounds reasonable enough, but many of these clauses then proceed to list examples of events that fall under the Force Majeure clause, and this is where things can unravel.

Some typical exclusions added in the Force Majeure clause include:

- Any events outside of the SaaS Provider’s control;
- Periods of emergency maintenance, for example for an urgent security patch;
- Failures that are caused by the SaaS Tenant, by action or inaction, or possibly another technology provider;
- Connectivity issues experienced by the SaaS Tenant

As a SaaS provider, you probably depend on a cloud provider for your infrastructure or a colocation provider hosting your own infrastructure. You also depend on an Internet provider or a Content Distribution Network (CDN). Even if you have a 100% uptime SLA with these providers, each of them will have a Force Majeure clause in their contracts. You will want to ensure that your own client contracts cover at least the same set of exceptions.

As a SaaS tenant, you will want to ensure that your provider’s Force Majeure clause doesn’t include hazards that should reasonably be already mitigated by the provider as part of the SaaS service. For example, hazards such as “accidents”, “inability to secure fuel, power and materials”, “mechanical breakdown” and “failure of third party systems” among others should raise some red flags in your contract review and warrant further explanations. By including events that should be reasonably mitigated by design of the service, the net effect is to render any SLA guarantee completely useless.

Additionally, many so-called "boiler plate" contractual clauses were conceived for an on-premise license contract. Applied verbatim to a SaaS contract, they may not reflect the expectations of a 21st century SaaS Tenant. Always be sure to read your entire agreement to ensure it reflects your requirements.

Below is an example of a Force Majeure clause embedded in the Salesforce Master Services agreement. In other contracts, you might find the Force Majeure language in a separate and dedicated clause.

> **3.1. Provision of Purchased Services.** We will (a) make the Services and Content available to You pursuant to this Agreement and the applicable Order Forms, (b) provide applicable SFDC standard support for the Services to You at no additional charge, and/or upgraded support if purchased, (c) use commercially reasonable efforts to make the online Services available 24 hours a day, 7 days a week, except for: (i) planned downtime (of which We shall give advance electronic notice as provided in the Documentation), and (ii) any unavailability caused by circumstances beyond Our reasonable control, including, for example, an act of God, act of government, flood, fire, earthquake, civil unrest, act of terror, strike or other labor problem (other than one involving Our employees), Internet service provider failure or delay, Non-SFDC Application, or denial of service attack.
>
> Source:
> http://www.sfdcstatic.com/assets/pdf/misc/salesforce_MSA.pdf

8.4 Escrow rights

A software provider will take all means possible to protect his intellectual property. For an on-premise customer, this means having access to an executable, but not to the source code. In the same manner, a SaaS customer will have access to the service but no access to the executable nor the source code.

If the SaaS Provider imposes his contract, this clause may not be present. However, in the case of enterprise software, both on-premise and SaaS, it is frequent request from the client's legal team.

Essentially, the SaaS Tenant is counting on the continued existence of the software provider and the continued support of the software package. In an on-premise situation, the customer does not have a copy of the software code, only the executables. If the software provider goes out of business or decides to abandon the product, the customer will be able to continue to run the executable but will no longer have access to updates, security patched and bug fixes. In the case of a SaaS solution, the situation is even worse since the client has no executable to run in-house.

An Escrow Clause obliges the SaaS Provider to place a copy of the source software with a trusted third-party. The Third-party custodian is instructed to deliver a copy of the source software to the SaaS tenant if the following events arrive:

- The SaaS Provider declares bankruptcy;
- The SaaS Provider ceases operations, or more specifically, ceases the ongoing support of the product line.

The Escrow Clause does not take effect if the SaaS Provider is bought by another company who continues the ongoing support of the service. Nor does it take effect if the SaaS Provider ends the support for an older version of the software as part of its upgrade policy.

Obviously, invoking an Escrow clause is not a panacea. It will not be without some difficulty that the SaaS Tenant will assemble the knowledge and talent required to recreate the service’s operating environment. So, it is definitely a measure of last resort.

8.5 Disaster Recovery

This is a clause that is often absent from the SaaS Provider’s base contract. A Disaster Recovery commitment consumes money and resources, but rarely generates revenue. So, most SaaS Providers try to avoid the commitment until they are forced to[15]. Compounding this is the false sense of security that many SaaS Tenants have concerning robustness and availability of the Cloud services that they consume.

SaaS Providers trying to sell “up-market” into large enterprise accounts may be forced to add these provisions to close the deal. Likewise, a SaaS Provider with no DR plan that is preparing for an IPO will need to disclose that his entire revenue stream may be at risk if a catastrophic event occurs.

For SaaS Tenants, most just assume that the service is “always on”. If the SLA for the service is Five 9s or Three 9s, the SaaS tenant will need to implement work-arounds for business-critical functions for those periods of downtime. What if the SaaS Service falls out of SLA because of a major data center fire or a natural disaster? If the SaaS Provider invokes his Force Majeure clause, the SaaS Tenant has little recourse but to wait until the service is restored.

Many B2C SaaS Providers and many Early stage B2B SaaS Providers offer inadequate disaster recovery protection in their base contracts. The following clause is from the Google Terms of Service. In lay terms, there is no commitment from Google as to the availability of a service such as Gmail.

> OTHER THAN AS EXPRESSLY SET OUT IN THESE TERMS OR ADDITIONAL TERMS, NEITHER GOOGLE NOR ITS SUPPLIERS OR DISTRIBUTORS MAKE ANY SPECIFIC PROMISES ABOUT THE SERVICES. FOR EXAMPLE, WE DON’T MAKE ANY COMMITMENTS ABOUT THE CONTENT WITHIN THE SERVICES, THE SPECIFIC FUNCTIONS OF THE SERVICES, OR THEIR RELIABILITY, AVAILABILITY, OR

ABILITY TO MEET YOUR NEEDS. WE PROVIDE THE SERVICES "AS IS". (45)

If the SaaS Provider is providing a business critical service, he should provide a serious disaster recovery plan. This plan should include the processes and infrastructure required to resume operations after a major disaster. The plan should also define what is considered a "disaster" and who invokes the disaster recovery process.

Other points that the SaaS Tenant should validate for business impact:

- The SaaS Provider may add limitations of liability in the event of a disaster, for example exoneration of any liability for loss of business revenue, loss of data or content, etc.
- The SaaS Provider may implement a DR plan which involves data center in another geographic location. This situation may invalid guarantees concerning data residency;
- The frequency and location of backups;
- The frequency and previous results of DR exercises.

9. SALES AND GO-TO-MARKET STRATEGIES

Running an on-demand company means abandoning many of the long-held tenets of software best practices and adhering to new principles.
– Byron Deeter

What are the factors that influence the Sales and go-to-market strategy of a SaaS company? How do you approach a consumer, a SMB and an enterprise market? What are the differences between a B2C and a B2B strategy?

In this section, we will present the notion of product/market fit. We will discuss different customer acquisition strategies such as free trials and "freemium" packages. And finally, we will discuss the economics of scaling your Sales team to achieve growth.

The top questions asked by SaaS entrepreneurs usually are:

- How fast should I be growing?
- How much cash should I burn?
- How do I scale?

Every business is different, but we will try to give insight that will help you answer those questions for your business.

9.1 How fast should I be growing?

True growth for a SaaS Provider is only possible once he has achieved product/market fit. The term product/market fit describes “the moment when a startup finally finds a widespread set of customers that resonate with its product.” (46). The product/market fit (PMF) concept was developed and named by Andy Rachleff (co-founder of Wealthfront, and of Benchmark Capital) and popularized by Eric Ries’ book Lean Startup. (46)

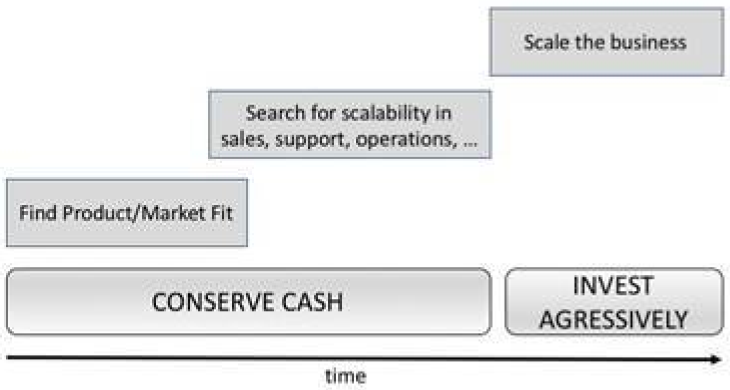
Marc Andreessen, of VC firm Andreessen Horowitz, offers this definition: “product/market fit means being in a good market with a product that can satisfy that market.” But too often the focus is on the product rather than the market. Andreessen emphasizes that market matters most: “You can obviously screw up a great market — and that has been done, and not infrequently — but assuming the team is baseline competent and the product is fundamentally acceptable, a great market will tend to equal success and a poor market will tend to equal failure.” (47)

Premature scaling is a huge pitfall for budding SaaS entrepreneurs. Early traction in the market by early adopters can often be misinterpreted as product/market fit. Ramping up the Sales team before all the right ingredients have been lined up can put a severe cash drain on the SaaS Provider, deepening the cash trough and moving it farther away from profitability. As I often say: “an overnight success usually takes between 5 and 7 years”. Market data tends to confirm this, with good companies taking an average over 7 years to reach 100M$ ARR, as illustrated in the following chart.

From $1M ARR to:	**Bottom 25%**	**Median**	**Top 25%**
Median years to reach $10M ARR	4	3	2
Median years to reach $100M ARR	10.6	7.3	5.3

figure 32 – BVP Growth Benchmark (48)

Real growth and scaling should only happen after you have found Product/Market Fit. Overall, your strategy should look like the one in the following figure.



Source: David Skok, SaaS Economics – Part 2: Scaling the business

figure 33 – Scaling the business

Brad Feld has introduced a metric which can be interesting (49). His proposition, dubbed the **40% rule**, is that your growth rate + your profit should add up to 40%. So, if you are growing at 30%, you should also be generating a profit of 10%. If your SaaS business is growing at a rate of 60%, you can be generating a loss of 20%. Feld's proposition is targeted for SaaS companies at scale, around $50M ARR and more.

For smaller companies, growth + profit can easily exceed 40%. For these earlier stage companies, a focus on the basic metrics, such as ARPA and CAC, that we presented in Chapter 4 may be more appropriate.

Marc Benioff describes his launch party with the B-52s: "At the time of the party, we had a small amount of revenue and were not profitable. I wasn't concerned about that, but not because this was during the go-go days marked by the "profits don't matter" mindset. I was confident that salesforce.com would be profitable." (5)

At the same time, Benioff was on the edge of triple digit growth. So, in all probability, Salesforce exceeded the 40% by a good margin.

As we will see in section 9.5, Valuations are often tied to growth rates, especially for early stage companies. Without significant growth, it is hard to get good valuations and harder to get financing.

9.2 How much cash should I burn?

The second question the SaaS entrepreneur asks is how much cash should I burn? We have seen in Chapter 4 that the annuity revenue model automatically creates a cash trough that can be fatal to the SaaS startup that burns too much cash, too fast.

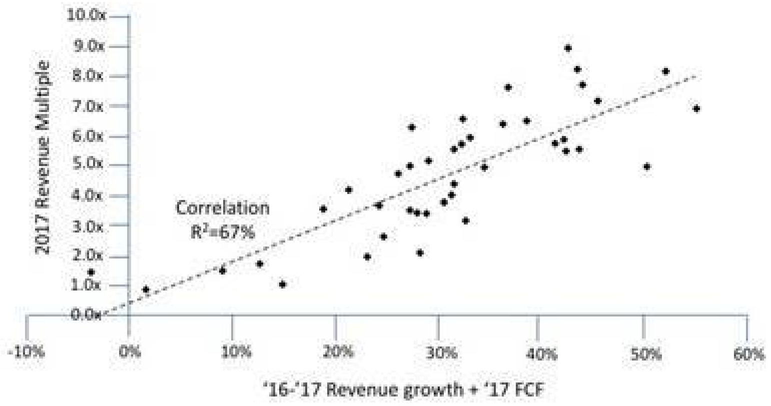
The BVP 2016 State of the Cloud Report (50) advances the proposition that although growth is an important metric, efficient growth is even more valuable. BVP measured the correlation between growth and the valuation of the 31 SaaS companies that compose their BVP Cloud Index. In 2013, the correlation between the two values, growth and valuation, was 69%; in 2016, the correlation was down to 39%. In other words, growth, by itself, is becoming less of an indicator of SaaS company value.

Current thinking is that investors are not just looking for growth, but for efficient growth. So instead of using the Rule of 40 as a key metric, BVP proposes an **efficiency score**.

$$\boxed{Efficiency\ Score = \%\ Annual\ CARR\ Growth + \%\ FCF\ Margin}$$

The Efficiency Score is based on measuring % revenue growth and adding % Free Cash Flow (or % Burn if you are still in the red). BVP found a correlation of 67% between Efficiency Score and Revenue Multiples (see figure 34).

However, the Efficiency index is not as easy to interpret as the Rule of 40%, since the Efficiency Score will be different at different stages of growth of the company. Usually, SaaS companies will have their best efficiency score just before their IPO, when they are growing fast and starting to generate profits. Typically, larger companies become less efficient as compared to smaller, nimbler, companies. As they mature, FCF will grow, but growth will slow.



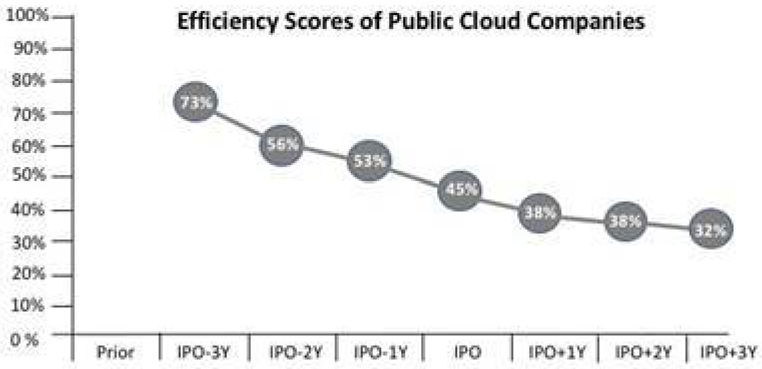
source: capIQ and BVP

figure 34 – Efficiency Score vs Revenue Multiple

The following graph illustrates the averages found by BVP, which we can summarize in three stages, as follows:

- Expansion ($30-60M ARR): 70%
- Pre-IPO ($60-100M ARR): 50%
- Public ($100M and more): 30%

Anytime your company manages to beat these averages, that means that you are doing great.



Source: BVP State of the Cloud 2016

figure 35 – Efficiency Scores over time

And what about early stage companies, in the zone labeled "prior" on the chart? What should their efficiency score be? For companies with less than $30M ARR, BVP proposes two metrics that should be more appropriate:

- Early stage (<$10M ARR): LTV / CAC > 3
- Mid stage ($10-$30M ARR): net new ARR / Net Cash Burn > 1

The first ratio, LTV / CAC, we discussed in Section 4.4.2.

The second ratio is essentially an indicator of cash flow health. If the ratio is greater than 1, then the SaaS Provider is slowly climbing out its cash flow trough. If the ratio is less than one, it is still sliding downwards towards increasing cash requirements.

9.3 How do I scale?

We discussed technical scalability in Chapter 5. In this chapter, we are talking about scaling the business: essentially investing more in sales and marketing to capture clients and market share. When should an early stage SaaS company start hiring more sales people? or hiring its first VP of Sales? These can be critical questions for if you scale too fast or prematurely, you burn cash and eventually fail.

> "In general, hiring before you get product/market fit slows you down, and hiring after you get product market fit speeds you up. Until you get product/market fit, you want to a) live as long as possible and b) iterate as quickly as possible." Sam Altman

One sign of potential product/market fit is that clients are signing up for your service in advance of the delivery of key features on your roadmap. In other words, the problem that you are trying to solve causes sufficient pain for customers that they are willing to forego some industry standard feature to be able to use your product right away.

Another potential sign of product/market fit is shortening sales cycles, in other words, sales people no longer have to educate the market, they just take orders.

Length of the sales cycle is one of the metrics that Benioff proposes to measure Sales Success (5). The others are:

- Inbound Sales
- Raw Web traffic
- Capture rate
- Lead conversion rate
- Close rate
- Average deal size
- % Business from new customers (as opposed to upsells)
- Sales cycle length

- Sales productivity (the average amount a sales person closes monthly)

By taking the lead conversion rate and close rate, you should be able to calculate how many raw leads you need in your sales funnel to channel to a sales person for him or her to reasonably meet his or her’s monthly sales target. If there are in insufficient amount of leads, chances are your new sales person will be unsuccessful.

By hiring more sales people, you will increase your ARR and growth rates, however, you will potentially increase the depth of your cash trough if you have not yet attained profitability. If you have all the signs of having achieved product/market fit, that there is no dominant player in the space yet, but you don’t have the cash to scale rapidly, then it is time for you to contemplate your next round of financing.

9.4 Customer Acquisition Strategies

To put this discussion on Customer Acquisition Strategies in perspective, let us revisit Christoph Janz's model for a 100M$ Company, published in his blog The Angel VC (51). Janz illustrates five scenarios to achieve $100M in ARR. Note that the X and Y axis are presented using a logarithmic scale.

Your customer acquisition strategy will vary whether your product and pricing are targeting 1000 elephants or 10,000,000 flies. In the market today, a SaaS Provider in the Enterprise space, such as Workday or Kinaxis, will target hundreds of customers at 1$ million per year each. At the end of the spectrum, SaaS Providers such as Github, Netflix and Spotify will target millions of customers at 10$/month.

Enterprise SaaS Providers, Workday and Kinaxis for example, can use real people to connect with their prospects; while low-cost high-volume Providers, such as Github and al. can only use automated customer sign-up and installation if they wish to scale to millions of users. Therefore, large enterprise SaaS solutions will use a "solutions selling" approach like what exists in the enterprise on-premise software market. We illustrate this "**high touch**" selling model in Figure 35.

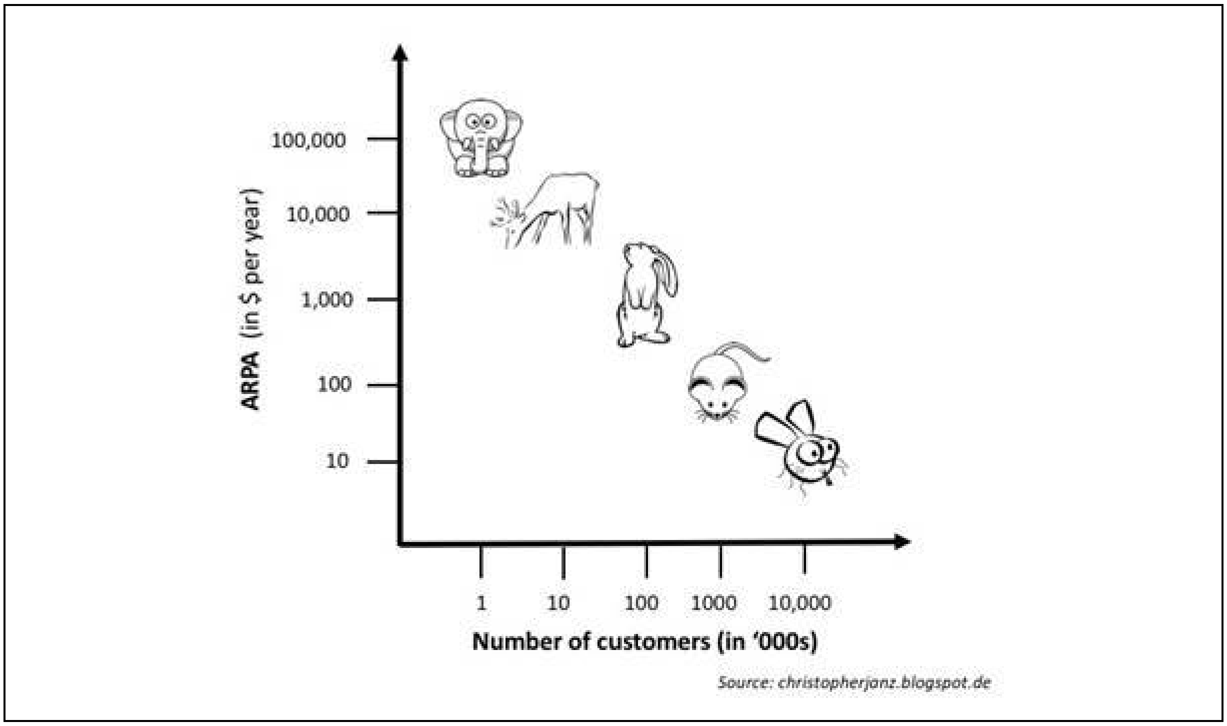
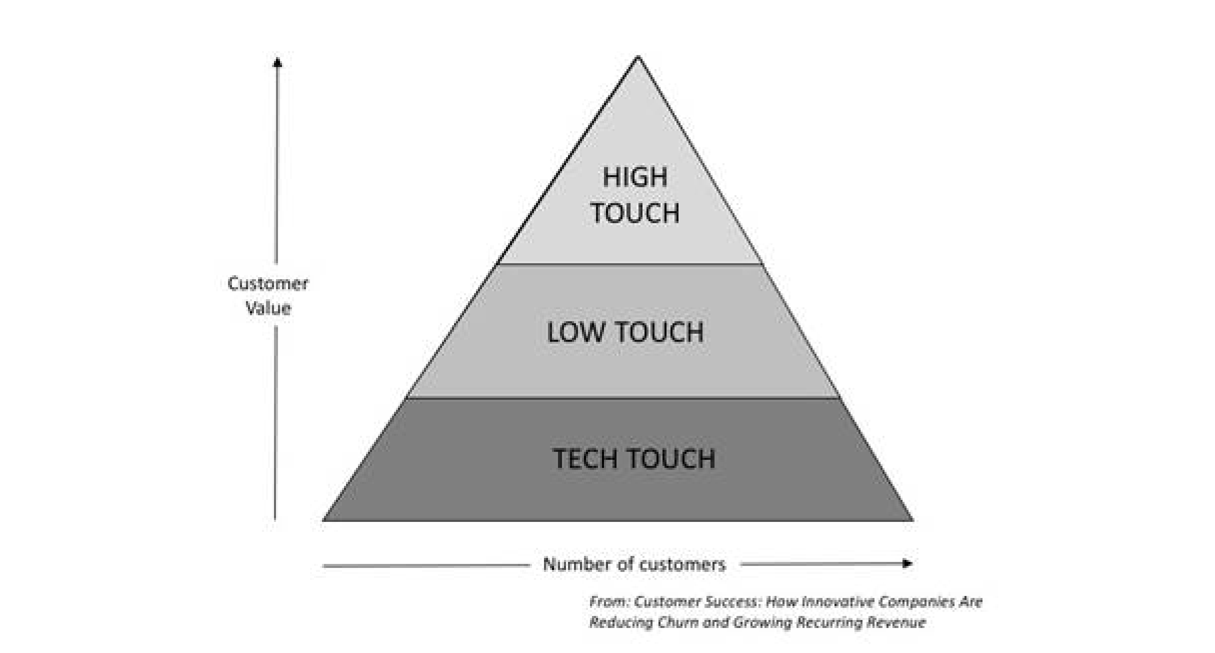


figure 36 – Five ways to build a 100M$ Business (51)

A "**tech touch**" approach to customer acquisition will translate into an easy to use self-serve sign-up process, such as the one used by Github, Netflix and others. Often all that is required is a valid email address to get access to a trial version of the service.

"**Low touch**" might be a hybrid solution, with major accounts getting personalized attention, but increased automation everywhere possible in the supply chain. For example, Salesforce's Revenue Recognition department operates "no touch" from quote to compensation for 75% of the over million orders that they handle every year. (52)



From: Customer Success: How Innovative Companies Are Reducing Churn and Growing Recurring Revenue

figure 37 – Different Customer Acquisition Strategies

Once the customer has signed up for a trial version, you will engage in a series of email reminders, or *pokes*, to incite him to sign up for a paid subscription. Several customer journeys can be mapped out in advance, based on the use the customer made of the app during his trial period. Tools such as Autopilot (http://autopilothq.com) can be used to automate the generation of email reminders and *pokes* in a tech touch scenario.

Another factor which will influence your sales approach is whether you are selling into an existing market or are you selling into a Blue Ocean. The former will command a market approach based on feature, functionality or pricing differentiation, while the latter will require more client education, content marketing, problem & solution awareness.

For example, ServiceNow was selling into a space already occupied by numerous other IT service desk solutions. They measured that service desk employees were spending about 20% of their time helping end-users get new passwords. They then crafted a campaign based on *“Passwords resets: Have you hit the boiling point. Take your temperature: <link>”.* Their campaign was successful and differentiated them from the competition since it offered personal value (password reset management) and connected with the target client on an emotional level as well (temperature = frustration).

Be conscious of your choices: Are you chasing elephants or rabbits? does your service require a high-touch or a low-touch sales approach? How much market education & awareness do you need to build into your plan? You can then start to craft your marketing & sales strategy, the object of which is to generate the qualified leads needed to grow the business. Simply put, you want to fill your buckets as full as possible, minimize leakage, and ensure maximum movement from one bucket to the next.

Content marketing in the following diagram is especially appropriate in a Blue Ocean market where customer education is warranted. This could take the form of White Papers, eBooks, Webinars, Case Studies, ROI calculators, etc. This may generate leads, but whether those leads qualify as potential customers can vary greatly.

Offering a Freemium or a “Try and Buy” service can be effective way to qualify visitors. Free trials are now prevalent in the SaaS industry, but are nearly impossible to do with on-premise enterprise software. This approach was pioneered by salesforce.com (who else?) back in 1999 when it was definitely anti-conformist. (5) Typically, a Freemium version is a version that is offered for an indefinite period but presents only a subset of the functionalities or imposes usage limitations compared to the full product. The bet is that enough Freemium users will be convinced of the advantages of the service that they will want to sign-up for the full version. If the Freemium version is wildly popular but has limited conversion rates, then CAC costs will rise with an obvious impact on the SaaS Provider’s burn rate. A “Try and Buy” version can be either a full-version or a limited functionality version that can be used for a fixed period of time, after which the service is shut down if the client doesn’t convert to a paid version.

To be successful, the sign-up process for a Freemium or Try and Buy should be as painless as possible. This will ensure a maximum number of sign-ups. However, adding some conditions, like registering a credit card, may be a valid way of qualifying the real interest of a trial user.

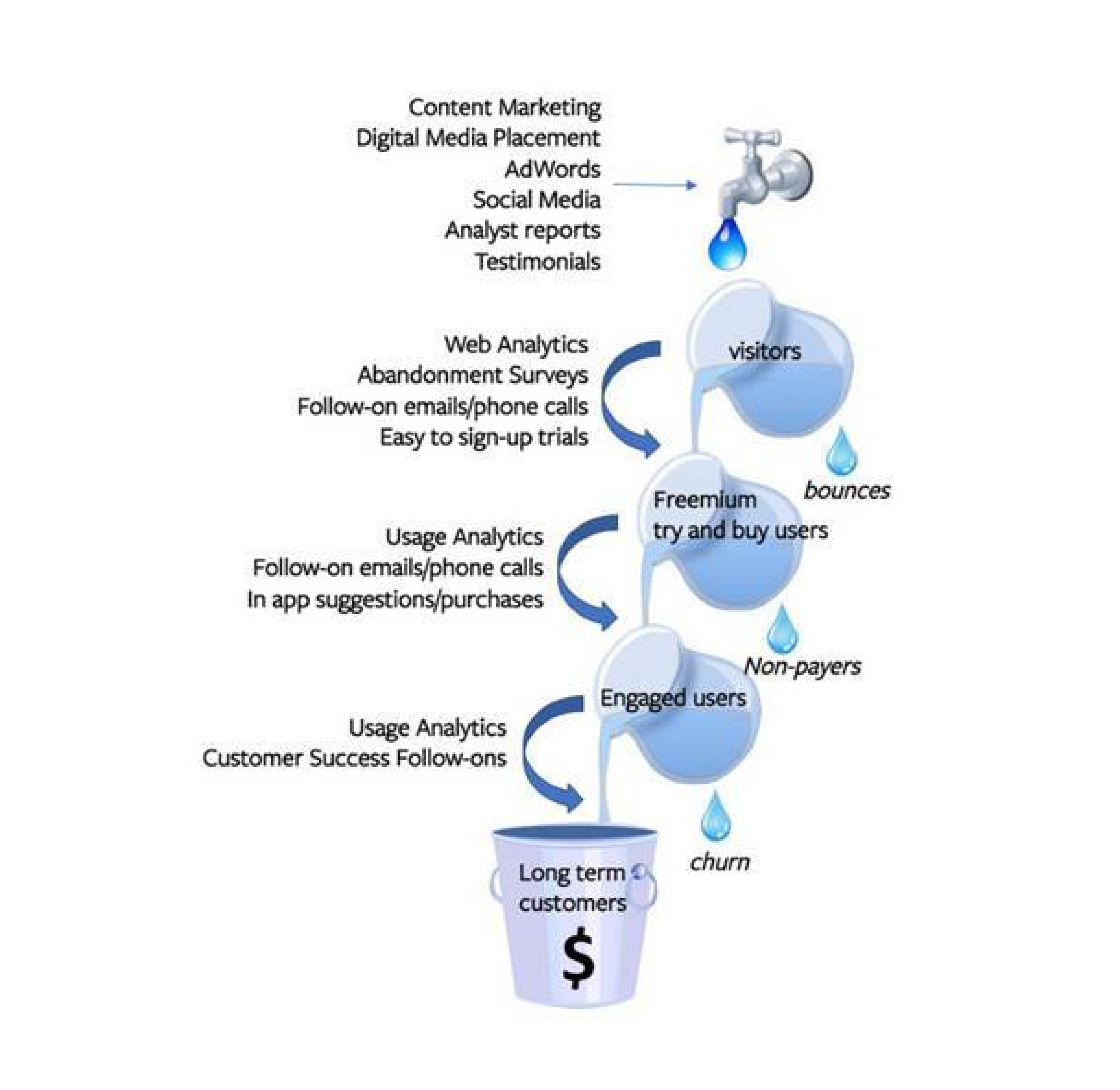


figure 38 – Filling the buckets

To manage this sequence in its entirety, the SaaS Provider’s Sales team needs to monitor cost per lead, sign-up rates, conversion rates at each bucket, usage levels, churn and other pertinent metrics

9.5 Valuations and getting funded

When we look at recent valuations, both from IPOs and from M&A, some numbers we can rationalize, while others seem to be a little bit of black magic. How are SaaS valuations derived exactly?

The following table gives the valuations for IPOs and M&A activity in recent years. It is likely that the people who subscribed to these IPOs or negotiated these acquisitions didn't all use McKinsey's standard recipe of Discounted Cash Flows (53). Even the multiples of Sales or Revenues don't always follow a set pattern.

Cloud IPOs	Acquisition Valuations
2015-2017	2009-2017
Atlassian $6.2B	LinkedIn $29.0B
Shopify $4.3B	Netsuite $9.3B
Twillio $2.9B	Concur $8.3B
Box $2.3B	Ariba $4.6B
Blackline $1.4B	Dealertrack $4.0B
Coupa $ 1.3B	SuccessFactors $3.7B
Mindbody $989M	AppDynamics $3.7B
appfolio 826M	Demandware $2.8B
Instructure 694M	Exact Target $2.6B
Apptio 592M	Fleetmatics $2.4B
Everbridge $492M	Skillsoft $2.3B
xactly 331M	Taleo $1.9B

Source: BVP State of the Cloud Report 2017

SaaS Capital, a specializing in alternate financing for SaaS companies, has done a credible exercise in determining SaaS company valuation. You can download their complete White Paper at: http://www.saas-capital.com/resources/wp-saas-company-valuation-st.

SaaS Capital first determines the revenue multiples of public SaaS companies since this data is public and available from multiple sources, such as the BVP Cloud Index (https://www.bvp.com/strategy/cloud-computing/index). Historical trends are around 7 to 3 X revenues (54).

SaaS Capital then subtracts 1.3 from that multiple to get a more representative figure for a private company.

Then SaaS Capital adjusts that multiple to account for growth. If your company is growing faster than average for its size, then it will be worth more than what you have calculated above; if it is growing slower than average, then it will be worth less.

In a final step, SaaS Capital considers four other metrics:

1. Size of addressable market
2. Retention Rate
3. Gross Margins
4. Capital Efficiency

If you score better than the market average on these metrics, it is influence your valuation positively. If you lagging behind the average, then your valuation goes down.

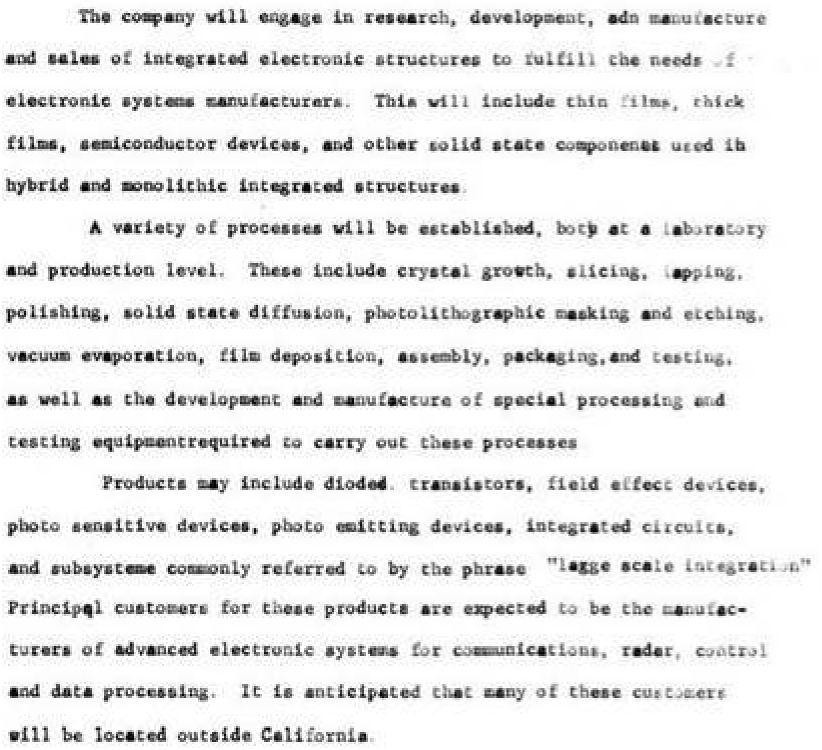
Obviously, your valuation is going to drive a lot of the funding discussions with potential investors. Point Nine a Berlin-based VC Fund, offers this framework for navigating the "funding roadmap":

	Pre-Seed	**Seed**	**Series A**	**Series B**	**Series C**
Amount Raised	$200K-$500K	$500K-$2.5M	$5-12M	$10-40M	$20M+
Valuation	$1-3M	$2-6M	$10-40M	$30-200M	$80M+
Investors	Friends&Family Angels	Angels Micro VC	VC	VC	VC Private Equity
MRR	-	$0-50K	$100-250K	$400K-$1.2M	$1M+

A Business Plan

How important is a solid business plan to succeeding your SaaS venture and in getting funding? If you are sitting through your MBA course on business strategy, say "essential". However, reality sometimes proves otherwise.

Take as an example Intel. Intel is the dominant player in the silicon chip market, with a market capitalization of over $170B and revenues of close to $60B. But back in 1968, it was just an idea in the head of two engineers disillusioned with their employer at the time, Fairchild Semiconductor. Intel's two founders, Robert Noyce and Gordon Moore, drafted a one-page business plan, complete with typo's, and presented it to pioneering Venture Capitalist Alan Rock who then used the plan to raise $2.5M in convertible debt. Thus, Intel was born and, as they say, the rest is history.



The company will engage in research, development, adn manufacture and sales of integrated electronic structures to fulfill the needs of electronic systems manufacturers. This will include thin films, thick films, semiconductor devices, and other solid state componenes used ib hybrid and monolithic integrated structures.

A variety of processes will be established, both at a laboratory and production level. These include crystal growth, slicing, lapping, polishing, solid state diffusion, photolithographic masking and etching, vacuum evaporation, film deposition, assembly, packaging,and testing, as well as the development and manufacture of special processing and testing equipmentrequired to carry out these processes

Products may include dioded. transistors, field effect devices, photo sensitive devices, photo emitting devices, integrated circuits, and subsysteme commonly referred to by the phrase "lagge scale integration" Principql customers for these products are expected to be the manufacturers of advanced electronic systems for communications, rader, control and data processing. It is anticipated that many of these customers will be located outside California.

source: Intel museum

figure 39 – Intel's original business plan from 1968

The brevity of the Intel business plan compared to their huge success is a testimonial that success is sometimes more a function of execution than the original idea itself.

The story of Fairchild Semiconductor, Intel, the birth of Silicon Valley and the birth of the Venture Capital industry is told in the 2011 film **Something Ventured**. In the film, the late Tom Perkins, a founder at Kleiner, Perkins, Caufield & Byers, one of the original Sand Hill Road VC firms, jokingly says “I don’t know how to write a business plan, I can only tell you how we read them. And we start at the back and if the numbers are big, we look at the front and see what kind of business it is.”

Behind the joke is a glaring truth: for most VCs, the technology doesn’t matter. No matter how much you have sweated over the elegance of the technology that supports your product, the investor will look first at the Management Team (which will be the initial reason for Tom Perkins to pick up your business plan in the first place) and second at the potential market. The business plan and the technology are secondary. To illustrate it another way, if you have two entrepreneurs named Bill Gates and Paul Allen, who both have IQs of 160, they could have built any number of successes other than Microsoft. It is the team that is important, not a venture in particular.

EPILOGUE

The Ten Laws of SaaS

Bessemer Venture Partners (BVP) , a global VC firm, was an early investor in many SaaS ventures that are now familiar names to all of us. BVP has invested and supported companies such as Skype, VeriSign, Yelp, LinkedIn, Pinterest, Box, DocuSign, Eloqua, Shopify and Twilio. Based on this experience, the partners at BVP developed a white paper called the Ten Laws of Being SaaS-y in 2009 (55).

Building on the success of that initial White Paper, BVP have produced several updates. Here is the February 10, 2016 version of their Ten Laws. (56) (57) After reading this book, hopefully many of these Laws will seem almost intuitively obvious.

Law 10: On Demand everything
Law 9: Grow (Efficiently) or Die
Law 8: Sales Efficiency is Your Efficiency
Law 7: Customer Success = Company Success
Law 6: Control Your Own Destiny…Cash is King
Law 5: 5C's of Cloud Finance

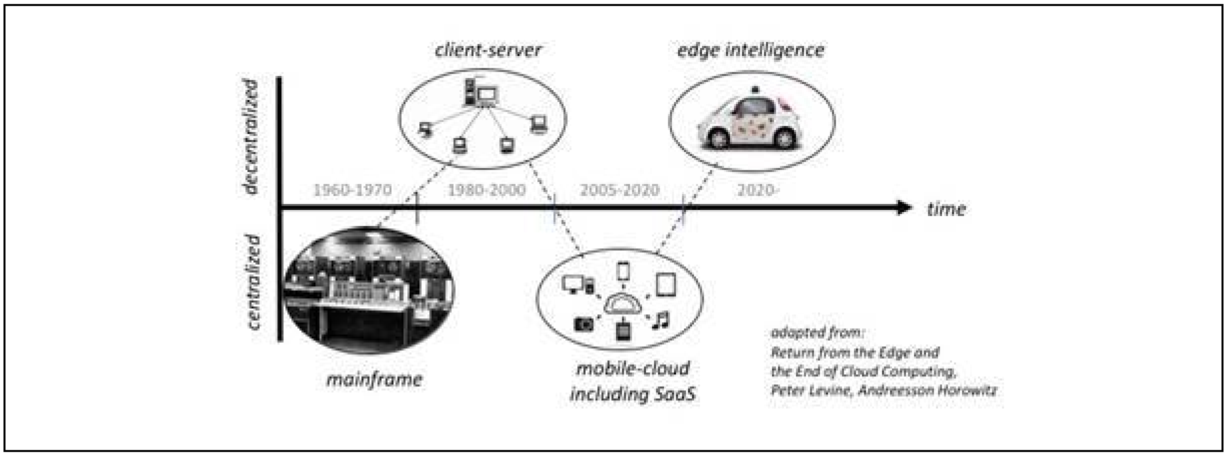
- **CARR & ARR**
- **CAC Payback**
- **Churn**
- **CLTV**
- **Cash Flow**

Law 4: The Best Product is Finally Starting to Win
Law 3: The Ascendancy of Developers
Law 2: Inspire (and Hire) with Vision
Law 1: Mobile is eating the World Wide Web

What’s next?

So what comes after the market shift to SaaS? It is a fatality that whatever technology is popular today will eventually get replaced. Peter Levine of the VC firm Andreesson Horowitz predicts the end of cloud computing as the world moves increasingly to the Internet of Things (IoT) (58).

As we have seen in Chapter 2, the software industry has evolved from a centralized computing model in the mainframe era to a decentralized or distributed model with client/server computing. We are now back to a centralized model with the Cloud and SaaS.



adapted from:
Return from the Edge and
the End of Cloud Computing,
Peter Levine, Andreesson Horowitz

figure 40 – Return to the Edge

In this mobile-cloud model, browsers, tablets, and smart phones all act as terminals used to access the centralized computing and storage resources in the Cloud. Witness digital assistants, such as Siri: no processing is done locally on the phone. The user’s vocal instructions are digitized and sent to a Siri server in a central hosting location for analysis and processing. A digitized vocal response is prepared at the central location, which is then sent to the user’s browser, tablet or phone.

In this mobile-cloud model, the “total addressable” market of users is approximately the earth’s population: a smart phone or a tablet for every human being on earth. So approximately 7.5 billion devices.

As we shift to “edge intelligence”, or what is commonly called the Internet of Things (IoT), we will have sensors in our running shoes, body cameras on police officers, lidars on self-driving cars, etc. Potentially trillions of devices collecting massive amounts of data, in real-time which, in certain cases, needs to be processed in real-time. This massive amount of data will force the edge (and not the center) to become more sophisticated with capabilities to process information and share information among peer devices.

For example, a self-driving car will contain hundreds of computers; it will literally become a data center on wheels. The real-time processing requirements, such as detecting a pedestrian or a stop sign, mean that processing will need to become localized.

Does that really mean the end of cloud computing? Probably not for a while. However, the role of cloud computing will shift to meet the requirements of the IoT. For example, data archiving, analysis of Big Data and meta-data to feed machine learning and AI processing will take place primarily in the Cloud.

Exciting times await us…..

Bibliography

1. **NIST.** *NIST Special Publication - The NIST Definition of Cloud Computing - 800.145.* Sept. 2011.

2. **Rightscale.** [Online] http://www.rightscale.com/blog/cloud-industry-insights/iaas-vs-paas-2015-cloud-trends-state-cloud-survey.

3. **NIST.** *NIST.* Sept. 2011. NIST Special Publication - The NIST Definition of Cloud Computing - 800.145.

4. **Council, Cloud Standards Customer.** *Practical Guide to Cloud Computing, Version 2.0.* 2014.

5. **Adler, Marc Benioff and Carlyle.** *Behind the Cloud.* [ed.] Josset-Bass. 2009.

6. **[Online] http://www.bloomberg.com/news/articles/2014-03-19/adobes-controversial-subscription-model-proves-surprisingly-popular.**

7. **[Online] https://qz.com/350272/adobe-saved-photoshop-by-ditching-boxes-for-the-cloud/.**

8. **Rennyson, Ahmed Bouzid & David. *The Art of SaaS.* s.l. : Xlibris US, 2015. B00ZCTS0HE .**

9. **Raymond, Eric S. *The Cathedral & the Bazaar.* s.l. : O'Reilly Media, 2001.**

10. **Warrillow, John. *Built to Sell: Creating a Business That Can Thrive Without You* . [ed.] Portfolio. 2011. 1591845823.**

11. **Nick Mehta, Dan Steinman, Lincoln Murphy. *Customer Success: How Innovative Companies Are Reducing Churn and Growing Recurring Revenue.* s.l. : Wiley, 2016. 978-1119167969.**

12. **Croll, Alistair. Pitchers not buckets. [Online] April 30, 2015. http://solveforinteresting.com/pitchers-not-buckets/.**

13. Campbell, Patrick. You're calculating CAC wrong. [Online] 11 23, 2016. http://blog.profitwell.com/youre-calculating-cac-wrong.

14. —. [Online] 10 20, 2015. http://www.priceintelligently.com/blog/how-to-get-your-saas-to-profitability-in-12-months.

15. Skok, David. www.forentrepreneurs.com/startup-killer/. *Startup Killer: the Cost of Customer Acquisition.* [Online] [Cited: 03 20, 2017.]

16. Harsh, Vardhane. *SaaS Startups for Beginners.* s.l. : V.H. Book Studio, 2016. B01D94GL1A.

17. baremetrics. baremetrics.com/academy/saas-quick-ratio/. [Online] [Cited: 03 20, 2017.]

18. Insight Squared. Why the Quick Ratio is a Crucial SaaS Metric. [Online] www.insightsquared.com/2016/02/quick-ratio-crucial-saas-metric/.

19. Tunguz, Tomasz. What is the Optimal Quick Ratio for your SaaS Startup ? [Online] 11 03, 2015. tomtunguz.com/what-is-quick-ratio-hiding/.

20. Paul Ferris, Neil Bendle, Philip Pfeifer, David Reibstein. *Marketing Metrics: The Manager's Guide to Measuring Marketing Performance.* s.l. : Pearson FT Press, 2015. 978-0134085968.

21. Securities, Pacific Crest. *2016 Pacific Crest Private SaaS Company Survey Results.* 2016.

22. —. *2015 pacific Crest Provate SaaS Company Survey Results.* 2015.

23. startupdefinition.com. DAU over MAU. [Online] http://startupdefinition.com/dau-over-mau .

24. Hamid, Mamoon. Numbers that actually matter. [Online] 02 09, 2017. https://www.slideshare.net/03133938319/numbers-that-actually-matter-finding-your-north-star.

25. ***Architectural Concerns in Multi-tenant SaaS Applications.*** **Rouven Krebs, Christof Momm, Samuel Kounev. 2012. Proceedings of the 2nd International Conference on Cloud Computing and Services Science.**

26. Platform, Google Cloud. Autoscaling Groups of Instances. [Online] https://cloud.google.com/compute/docs/autoscaler/.

27. Barroso, Luiz and Hölzle, Urs. ***The Datacenter as a Computer.*** **s.l. : Morgan & Claypool, 2009.**

28. Morgan, Timothy Prickett. A Rare Peek Into The Massive Scale of AWS. [Online] 11 14, 2014. https://www.enterprisetech.com/2014/11/14/rare-peek-massive-scale-aws/.

29. Netflix. Netflix Chaos Monkey Updated. [Online] 10 19, 2016. http://techblog.netflix.com/2016/10/netflix-chaos-monkey-upgraded.html.

30. Magazine, Wired. The Epic Story of Dropbox's Exodus From the Amazon Cloud Empire. [Online] 3 14, 2016. https://www.wired.com/2016/03/epic-story-dropboxs-exodus-amazon-cloud-empire/.

31. Whitepaper, AWS. ***SaaS Solutions on AWS: Tenant Isolation Architectures.*** **2016.**

32. Internap. Performance IP Service. [Online] http://www.internap.com/network-services/performance-ip/.

33. Atchison, Lee. ***Architecting for Scale.*** **s.l. : O'Reilly, 2016. 9781491943397.**

34. Commerce, Office of Government. ***ITIL Service Strategy.*** **s.l. : The Stationary Office, 2007. 978-01133110456.**

35. ISACA. Control Objectives for Information and Related Technologies. [Online] http://www.isaca.org/cobit/pages/default.aspx.

36. Google. GOOGLE CLOUD PLATFORM SECURITY. [Online] https://cloud.google.com/security/compliance.

37. Pham, Thu. SOC 1, SOC 2, SOC 3 Report Comparison. [Online] 8 19, 2011. http://resource.onlinetech.com/soc-1-soc-2-soc-3-report-comparison/.
38. AICPA. Comparison of SOC 1, SOC 2 and SOC 3 Reports. [Online] [Cited: 3 27, 2017.] https://www.aicpa.org/InterestAreas/FRC/AssuranceAdvisoryServices/DownloadableDocuments/Comparision-SOC-1-3.pdf.
39. NDB. SOC 1 vs. SOC 2 Understanding the Key Differences & Similarities and What You Need to Know. [Online] [Cited: 3 27, 2017.] https://socreports.com/white-papers/soc-1-vs-soc-2.html.
40. Commission, Comittee of Sponsoring Organizations of the Treadway. [Online] www.coso.org.
41. Alliance, Cloud Security. [Online] cloudsecurityalliance.org.
42. *Cloud Computing and the USA Patriot Act: Canadian Implications.* Banks, Timothy M. 3, July 2012, Internet and E_Commerc Law in Canada, Vol. 13.
43. Group, Object Management. Data Residency Working Group. [Online] [Cited: 03 13, 2017.] http://www.omg.org/data-residency/.
44. Netsuite. www.netsuite.com/termsofservice. *Terms of Service (Revised Jan 2017).* [Online] [Cited: 03 08, 2017.]
45. Google. www.google.com/intl/en/terms/. [Online] [Cited: 03 09, 2017.]
46. Ries, Eric. *Lean Startup:* . s.l. : Crown Business, 2011. 978-0307887894.
47. Andresson, Marc. [Online] https://www.youtube.com/watch?v=zfOsP3PmI1U .
48. Partners, Bessemer Venture. State of the Cloud Report 2017. [Online] www.bvp.com/cloud.
49. Feld, Brad. The Rule of 40% For a Healthy SaaS Company. [Online] Feb 3, 2015.

http://www.feld.com/archives/2015/02/rule-40-healthy-saas-company.html.

50. Byron Deeter, Kristina Shen. BVP State of the Cloud Report 2016. [Online] www.bvp.com/cloud/.

51. Janz, Christoph. The Angel VC. [Online] 10 5, 2014. chrostophjanz.blogspot.de/2014/10/five-ways-to-build-100-million-business.html.

52. Smith, Meredith. Revenue Ops: The Salesforce Way. [Online] https://www.salesforce.com/form/appexchange/revenue-ops-the-salesforce-way.jsp .

53. Tim Koller, Marc Goedhart, David Wessels. *Valuation: Measuring and Managing the Value of Companies.* [ed.] McKinsey & Company. Sixth Edition. s.l. : Wiley, 2015.

54. Tunguz, Tomasz. The 57% Drop in SaaS Valuations. [Online] February 7, 2016. http://tomtunguz.com/depression-in-saas/.

55. Deeter, Byron. Bessemer's Top 10 Laws for Cloud Computing. [Online] November 10, 2009. http://sandhill.com/article/bessemers-top-10-laws-for-cloud-computing/.

56. Byron Deeter, Kristina Shen, Umair Akeel. Bessemer's 10 Law's of Cloud Computing. [Online] February 10, 2016. https://www.bvp.com/blog/bessemers-10-laws-cloud-computing.

57. Partners, Bessemer Venture. The 10 Laws of Building a Successful SaaS Company. [Online] 10 27, 2008. https://www.slideshare.net/botteri/bessemer-10-laws-of-being-saasy-fall-2008-presentation.

58. Levine, Peter. Return to the Edge and the end of Cloud Computing. [Online] 12 16, 2016. [Cited: 01 23, 2017.] http://a16z.com/2016/12/16/the-end-of-cloud-computing/.

59. NIST. https://www.nist.gov/programs-projects/cloud-computing. *https://www.nist.gov/programs-projects/cloud-computing.* [Online] https://www.nist.gov/programs-projects/cloud-computing.

60. http://www.cloud-council.org/deliverables/CSCC-Practical-Guide-to-Cloud-Service-Agreements.pdf. [Online]

61. Commerce, Office of Government. *ITIL Service Strategy.* s.l. : The Stationar .

62. Gardner, Todd. 2017 Private SaaS Valuation Data. [Online] January 17, 2017. https://www.saastr.com/2017-private-saas-valuation-data-and-why-your-company-is-probably-not-worth-10x-revenue/.

ABOUT THE AUTHOR

Robert Michon is an independent advisory in technology, specializing in Software as a Service and the colocation industry. He has over thirty-five years of experience in IT, having spent the earlier part of his career with a major consulting organization. He holds a B.Sc.(Hons) degree in Computer Science from the University of Manitoba, a M.B.A. from Laval University and a Certificate in Global Management from INSEAD Business School.

In 2003, he joined Taleo as Vice-President of IT services. Taleo was one the early leaders of SaaS-based Human Capital Management, growing at 100% year over year. This early experience in managing and scaling a SaaS service led to ongoing consulting assignments with both start-ups and well established software houses in the SaaS space.

In 2016, he co-founded SaaS360.io to productize several of the SaaS management tools that he conceived and developed at various companies.

When he is not tied to a keyboard at his home office in Quebec City, he can be found on a bike or a pair of skis somewhere in the world.

---

[1] More on DevOps in Chapter 3.

[2] Burroughs, UNIVAC, NCR, Control Data Corporation (CDC), Honeywell, RCA and General Electric,

[3] And as history has a way of repeating itself, Microsoft was the object of an anti-trust action for bundling its browser Internet Explorer in the Windows operating system. **United States v. Microsoft Corporation 253 F.3d 34**

[4] Business to Business, Business to Consumer, Business to Government

[5] An alternate description of Cloud Service Provider roles can be found in the ISO/IEC 17789 standard.

[6] Pacific Crest's 2016 SaaS Company Survey indicates that indeed Professional Services revenues are a minor component in a SaaS company's revenue stream. The median value is Services represents 16% of the first year contract value and the median margin on those services is only 22%. (21)

[7] For example, in a Java environment, each application instance would be instantiated by a Java Virtual Machine (JVM).

[8] For a more detailed understanding of Cloud architecture and design, refer to documents discussing RESTful services. The concept of REST was developed by Roy Fielding: https://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm

[9] In a similar fashion, slow response time or a sluggish application can also be considered, by the end-user, to be "unavailable".

[10] Formerly marketed as PNAP.

[11] Starting on May 1, 2017, SSEA audits ar e governed by the application of SSAE-18. This new AICPA Standard replaces SSAE 16, just as SSAE 16 replaced SAS 70.

[12] IXmaps is a project lead by the University of Toronto and York University that offers Internet users a tool that will map their own traffic. Over the past year IXmaps has crowdsourced traffic routing data from individuals that have contributed about 325,000 traceroutes. Based on that data, it estimates that 90 per cent of Canadian Internet traffic is being routed through the U.S. See www.ixmaps.ca

[13] This section is meant to explain SaaS contracts in lay terms and is not meant to provide legal advice. For legal advice, always refer to a qualified lawyer or attorney.

[14] Some service contracts may appear in two sections: A Master Services Agreement (MSA) that contains the general service obligations and one or more Transaction documents that specify the specific services and quantities ordered.

[15] In more rare occasions, SaaS Providers design their service to be stateless and geographically redundant so that disaster recovery is automatically built into the service.