

# Migrating Applications to the Cloud

An Amazon Web Services Case Study



**HyperStratus**

Virtualization, Cloud Computing ... and more

A HyperStratus White Paper

## Table of Contents

Executive Summary .....	2
Overview of SVEF and Lessonopoly .....	3
Lessonopoly Description .....	3
Lessonopoly Architecture .....	4
Motivation for Considering Cloud Computing .....	5
Limited System Scalability .....	5
Limited Storage Scalability .....	5
Vulnerability to Hardware Failure .....	5
Cost .....	6
Evaluating Cloud Computing .....	6
AWS Architecture .....	7
Project Migration to AWS .....	9
Key Cloud Computing Migration Issues .....	10
Software Licensing .....	10
Dynamic Data Management .....	11
Application and System Management .....	13
Application and System Backup .....	15
Lessonopoly AWS TCO .....	16
Scenario 1: Single Server .....	17
Scenario 2: Multi-tier .....	18
Scenario 3: Multi-machine, multi-tier .....	19
Lessonopoly TCO Analysis .....	20
Factors affecting TCO .....	20
Conclusion .....	22
About HyperStratus .....	23



## Executive Summary

The Silicon Valley Education Foundation serves 13,500 teachers and 260,000 students throughout Santa Clara County, located in the heart of Silicon Valley. To help teachers collaborate and enable peer learning, SVEF created Lessonopoly, a Web 2.0 application that offers lesson plan creation, modification, and sharing, thereby fostering improved education. A key feature of Lessonopoly is its round-the-clock accessibility, making it easy for teachers to work on lesson plans at their convenience.

The infrastructure that Lessonopoly was originally deployed upon offers no redundancy, exposing SVEF to the risk of an extended outage should any hardware fail. After examining the option of deploying duplicate hardware along with high availability virtualization, SVEF decided to pursue a different option: moving Lessonopoly into the cloud.

The application migration was highly successful, with the result of better application robustness. In addition, monthly hosting charges were reduced by moving the application into the cloud. This whitepaper discusses the migration process for Lessonopoly, including:

- Motivations for considering cloud computing
- Technical aspects of the effort
- Key cloud migration issues
- Lessonopoly migration project TCO



## Overview of SVEF and Lessonopoly

The Silicon Valley Education Foundation (SVEF) supports education throughout the Santa Clara Valley, a large and diverse collection of school districts. Serving 34 school districts, with 13,500 teachers and 260,000 students, SVEF delivers a range of programs, including:

- Teacher grants to support supplemental learning opportunities
- Intensive tutoring to students to enable greater success rates
- A Science, Technology, Engineering, and Math program designed to give students the skills necessary for tomorrow's jobs in Silicon Valley

### Lessonopoly Description

As part of its efforts to enable high-quality teaching, SVEF recognizes it needs to help teachers with technology-based solutions. After extensive research, SVEF concluded a valuable initiative would be to assist teachers with their lesson plans, which are the building blocks of individual class sessions. In the past, lesson plans have been paper-based, with no ability to contain or link to rich data sources that can be used in individual lessons. For example, video can be a rich resource for teaching many subjects, but paper-based lesson plans can, at best, list the online location of a relevant video. Furthermore, paper-based lesson plans are a poor medium for collaboration, making it difficult for experienced teachers to share knowledge and best practices with less-experienced colleagues.

Therefore, SVEF developed Lessonopoly, a Web 2.0 application facilitating lesson plan creation, modification, and sharing. Lessonopoly offers the ability to link rich data sources to lesson plans; these data sources may be stored within Lessonopoly itself or on another server located on the Internet. Lessonopoly offers users the ability to rate and comment upon individual lesson plans, thereby offering user-based quality control. In addition, it offers search functionality to allow users to seek lesson plans by title, content, or description. Finally, Lessonopoly allows creation of new lesson plans by editing one or more existing lesson plans and saving the updated document as a new lesson plan. This facilitates teacher customization of material to suit the needs of their classes.



## Lessonopoly Architecture

Lessonopoly is a web-based content management system based on the open source product Drupal. Lessonopoly is a three-tier application, with user interaction being funneled through the Apache webserver to webpages served up by the Drupal engine, which in turn accesses a MySQL database where Drupal configuration and page definition data is stored.

As originally designed, Lessonopoly's three tiers all run on a single server. Upon launch, low user volumes made it possible to handle all load on one server. Drupal-based systems are very flexible in terms of deployment, making it easy to spread the individual tiers onto additional servers as load increases. Some options for addressing Lessonopoly system load and robustness in a cloud environment are discussed later in this paper.



# Motivation for Considering Cloud Computing

As just noted, the original deployment topology of Lessonopoly was a three-tier system, all of which resided on the same machine, a 1U server hosted in a colocation hosting provider. While the hosting company is reputable and offers responsive service, a number of drawbacks exist with the arrangement.

## Limited System Scalability

The single server that Lessonopoly ran on was limited in its ability to scale to meet potential traffic. Meeting potential scalability requirements would require purchase of additional servers, something SVEF wanted to avoid, particularly in light of the current economy. Moreover, since user load volumes are variable, purchasing enough hardware to meet peak demand would result in excess hardware during low-demand periods, thereby tying up capital in unneeded capacity.

## Limited Storage Scalability

Lessonopoly offers the ability for teachers to upload large documents and digital media like audio and video, which require large amounts of storage. Since individual servers are typically limited to two drives, it is likely that Lessonopoly will eventually require more storage than is possible on a single server. One option to address this is to move system data to a specialized storage device, which can contain more drives.

However, this would require additional hardware purchase and poses the same issue of over-provisioning capacity. This would have the effect of burdening SVEF with excess capacity until the amount of Lessonopoly storage grew sufficiently to take advantage of the storage device.

## Vulnerability to Hardware Failure

Lessonopoly serves as a hub for teacher activity in lesson plan sharing and collaboration. It is critical that it be available at all times; system outages could hamper teachers in their ability to deliver education.



# Motivation for Considering Cloud Computing

---

In its original configuration, Lessonopoly was installed on a single server. This poses a risk, since hardware failure could result in system unavailability until repairs were made. If a component like a network card or a motherboard were to fail, a day or more could pass before system availability was restored. If a disk drive were to fail, not only would the device need to be replaced, but the system would need to be restored from backup, taking even longer.

This vulnerability would be worse if system load required additional hardware to enable the application tiers to be spread across multiple servers. Each server poses hardware failure risk at its level, which impacts total application availability; in essence, moving to a multi-tier hardware topology raises overall system risk.

## Cost

In its original configuration, hosting fees for Lessonopoly were \$200 per month, a relatively modest cost structure. However, as previously noted, the original application topology posed unacceptable risk to system downtime due to hardware failure. SVEF engaged HyperStratus to evaluate the Lessonopoly architecture and propose an alternative to reduce downtime risk.

One option examined was using virtualization software to implement high availability on redundant hardware. This effectively would address the risk of downtime, since any hardware failure would cause the virtualization management software to shift virtual machines to other hardware.

While use of virtualization software would address downtime risk, it would impose a significant cost. At least one, and possibly three or four new servers would be required to implement this topology, each costing around \$2000. Finally, an additional hosting fee of \$200 per month would be required for each new server.

## Evaluating Cloud Computing

Because reducing Lessonopoly risk via virtualization and additional hardware would cost more than SVEF wanted to spend, HyperStratus and SVEF agreed to evaluate whether a cloud implementation would be appropriate for Lessonopoly. Amazon Web Services (AWS) was selected as the target cloud implementation to evaluate.

Because AWS, as part of its internal architecture, uses virtualization, SVEF would be shielded from the risk of hardware failure. In addition, should individual instances of the SVEF application crash, restart is easy and therefore downtime is kept to a minimum.



## AWS Architecture

Amazon describes AWS as “an infrastructure web services platform in the cloud. With AWS you can requisition compute power, storage, and other services – gaining access to a suite of elastic IT infrastructure services as your business demands them.”

AWS can be characterized as “Infrastructure as a Service.” This means that Amazon provides basic computing capability – a virtual machine container, reliable and redundant storage, and high performance networking – in a remote location. Users have no need to provision or pay for local hardware infrastructure – Amazon takes care of that. Users focus on the software assets – the application – that resides upon and uses AWS computing resources.

AWS is comprised of a number of individual services; the key services for the needs of Lessonopoly are these four:

- Elastic Compute Cloud (EC2): EC2 provides elastic computing capacity. In essence, EC2 provides empty virtual machines into which users install desired software assets: operating system, middleware, and application(s). EC2 instantiates the collection of these assets as an “Amazon Machine Image.” Users can create their own AMIs from scratch; use pre-built AMIs available from the AWS website; or use a pre-built AMI as a starting point and then add additional software assets to finalize the desired AMI.
- Simple Storage Services (S3): S3 provides the ability to store any amount of desired data in a reliable fashion. It’s important to understand that S3 data is unstructured – S3 data is a “bag of bits.” It is up to the end user to impose structure on an S3 data collection. EC2 AMIs are stored in S3 and are instantiated from their S3 location. S3 data is not only unstructured, it is not updated dynamically. Updating an EC2 AMI requires a full “burn” process and results in a second AMI, somewhat different than the original one.



- Elastic IP Addresses: A feature of AWS is that DNS addresses for running EC2 instances are dynamically assigned at runtime. For web-based applications that are user-facing, AWS offers “Elastic IP Addresses” that are permanently assigned to a particular AMI, available at a small charge. Elastic IP Addresses are rationed by Amazon; therefore, it is important for applications that have EC2 instances that run without end user interaction to register the dynamic IP addresses with a known DNS server to enable inter-application EC2 instance communication.
- Elastic Block Storage: AWS also offers the Elastic Block Storage Service (EBS) which is persistent storage that is attached to running EC2 instances. With EBS, applications can save data permanently without needing to burn an entire new AMI.



## Project Migration to AWS

Lessonopoly is a Linux-based application; there are many Linux-based AMIs available on the AWS website. However, in order to ensure the security of SVEF's system, rather than use a Linux AMI of unknown provenance, HyperStratus created a new AMI.

The first step in the creation process was to install and configure the necessary software on a local physical server. New copies of CentOS, XAMPP (an aggregation of software including Apache, MySQL, and the Perl and PHP languages), and Drupal were obtained and installed.

The next step involved uploading and converting the local system into an AMI image. Amazon provides a suite of tools to accomplish this task. The process of creating what Amazon refers to as a "bundle" results in an object that can then be uploaded as an AMI stored within S3. Once this AMI is created, it may be registered with AWS and is then ready to run. The process of creating an EC2 instance from a local system is shown in Figure 1.

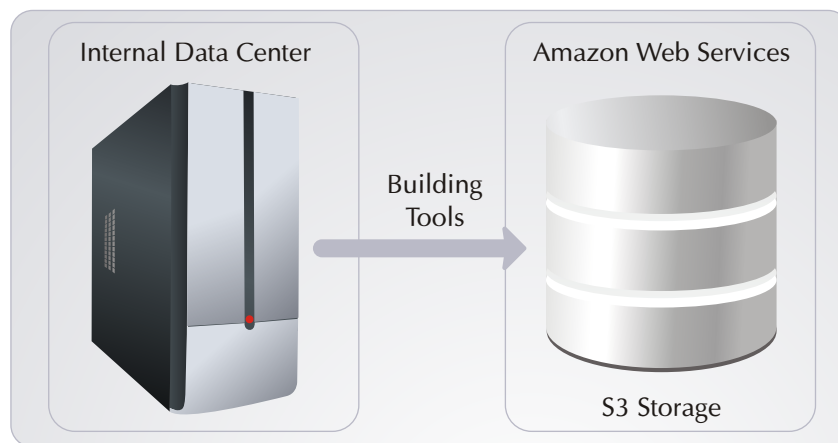


Figure 1

A third step involved creating a permanent IP address for Lessonopoly in AWS. An Elastic IP Address was obtained and assigned to the Lessonopoly AMI, thereby enabling consistent access via a domain name rather than requiring an ever-changing IP address, dynamically assigned each time Lessonopoly was started in AWS.



## Key Cloud Computing Migration Issues

Once these three steps were performed, the Lessonopoly AMI was ready to run. Notwithstanding the relatively straightforward migration process, a number of issues present themselves in a typical migration process that must be understood and addressed. These issues include software licensing, management of dynamic data, system monitoring and management, and system backup.

### Software Licensing

The Lessonopoly migration process was aided by the fact that the application is built entirely from open source software. This meant that no additional commercial licenses were needed during the initial system build. Furthermore, since open source licenses impose very few restrictions in terms of use, it was easy to run all the necessary software components in the Amazon cloud environment.

The terms of commercial software licenses are typically more restrictive, and may not allow deployment in a cloud environment. Moreover, many software vendors do not offer support for their products used in a cloud environment.

The issue of software licensing is likely to be a common challenge for many companies in their use of cloud computing. Most commercial software is licensed on a perpetual basis with a per-server licensing basis. No provision for transitory use with scaling across varying numbers of server instances is made. As organizations deploy cloud-based applications, matching system operational characteristics with commercial software licensing conditions will pose a challenge.



## Dynamic Data Management

As noted earlier, the default behavior of AWS is that AMIs are stored in S3 and instantiated in EC2 when made operational. No dynamic storage capability is provided. Any changes made to a running EC2 instance are not automatically stored in the original S3 AMI; in effect, an AMI is a “gold master” that remains unchanged even if modifications are made to a running instance spawned from the original AMI. To permanently store new or modified data, a new AMI must be “burned” and stored; that leads to AMI sprawl as fresh AMIs must be created each time an AMI is shut down with changed data contained within it. The process of launching, running, and then “burning” the updated EC2 instance is shown in Figure 2.

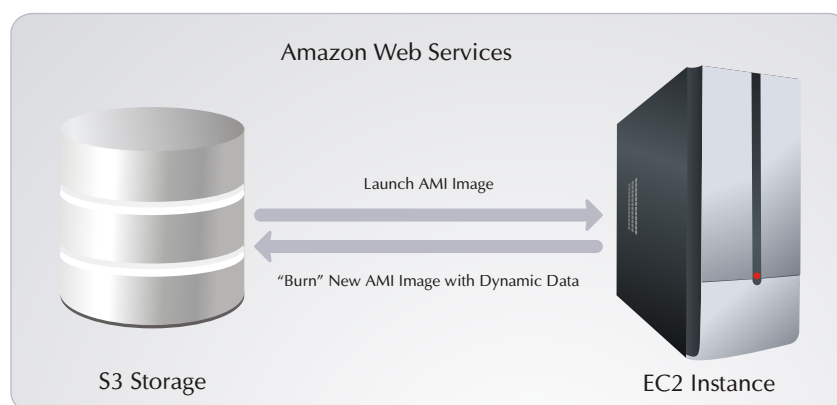


Figure 2

One common practice to address this “gold master” issue is to capture changed files within an EC2 instance and save them to a separate S3 storage object. An example of changed files might be individual html files (i.e., web pages). If an EC2 instance has new html files that need to be available when it runs, rather than burning a new AMI, the html files are stored to S3 prior to EC2 instance shutdown. Subsequently, when the AMI is brought up, the separate S3 object is imported to the now-running EC2 instance and applied to its files. In this fashion, dynamic changes to an EC2 instances can be accommodated.



# Key Cloud Computing Migration Issues

This process is illustrated in Figure 3. This “gold master” plus incremental change incorporation process enables an application to incorporate dynamic data without requiring repeated complete AMI “burning.” It does, however, impose some application complexity in that new S3 storage objects are created prior to each EC2 instance shutdown, thereby increasing the complexity of configuring and managing an application.

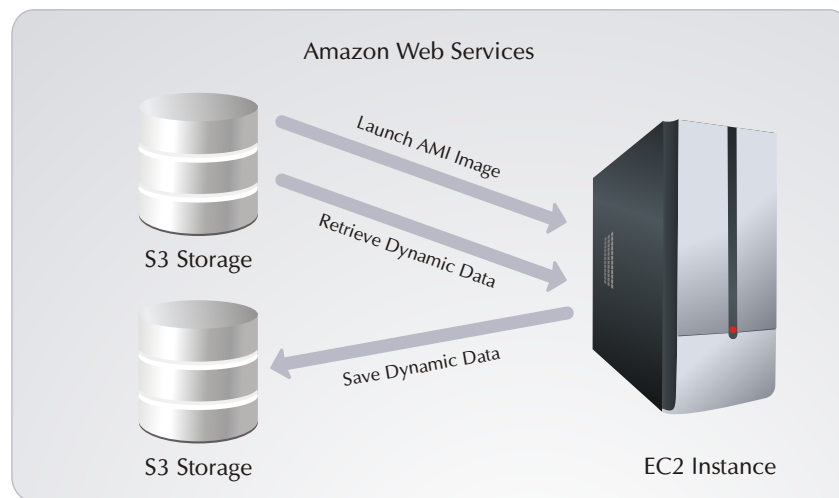


Figure 3

While the practice of separating dynamic data from base AMI can successfully accommodate dynamic data, it is less than satisfactory for applications that have large amounts of dynamic data, which tends to overload the flexibility of the “gold master plus incremental S3 storage object” approach. In addition, it is not appropriate for applications which require persistent and reliable SQL databases that can be recovered after a system failure. Lessonopoly is an example of an application that does not map very well into this approach, as it is a very dynamic system. Drupal, which forms the heart of Lessonopoly, uses a MySQL database to store information about new users who register. As lesson plans are created or modified, they are stored, with metadata (e.g., creation date, originating user) placed into Drupal, with associated documents or digital assets like video stored elsewhere in the Linux file system. In short, the application is constantly changing.

Rather than applying the “gold master plus incremental object” approach, Lessonopoly on AWS instead uses the Elastic Block Store (EBS) Service that Amazon provides. EBS enables assignment of all or part of an AMI’s file system to a service that reliably stores dynamic data.

A common use of EBS is to map relational databases to EBS, enabling AWS-hosted applications to mirror common application architectures, which use a relational database for reliable data storage and queries. Non-database files and directories that hold dynamic data in the AMI’s file system can also be mapped to the EBS resource.



Figure 4 illustrates how Lessonopoly uses EBS as part of its AWS architecture.

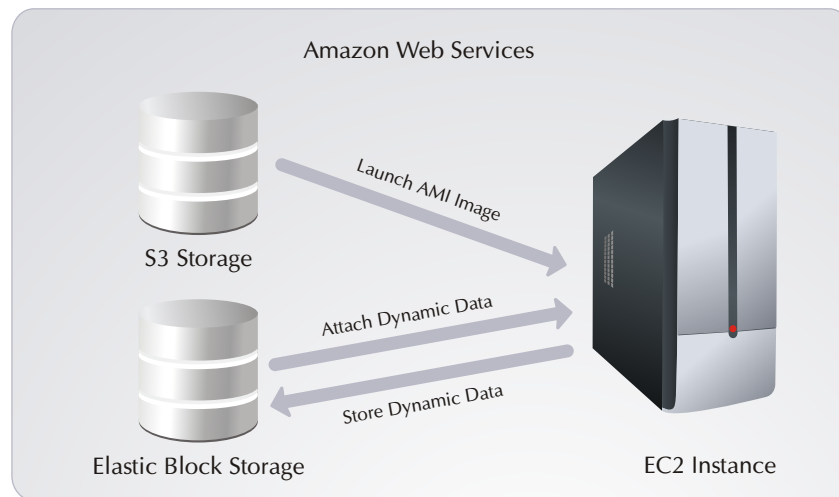


Figure 4

EBS resources may only be attached to running AMI instances; any connection between an EBS resource and a specific AMI is lost when that image is stored in S3. This means that the EBS resource must be attached upon AMI startup. With Linux-based AMIs, this is typically implemented by mounting the EBS resource to a directory within the AMI's file system. (Note that EBS resource, must be formatted prior to initial use; otherwise, attempting to mount the resource onto the AMI file system will fail). This mount process is accomplished via a script that is placed in the system bootup routine; each time the AMI is brought up, the script is executed, which attaches the EBS resource to the Lessonopoly file system, where system changes are stored persistently.

By using EBS, Lessonopoly can use the MySQL-based Drupal system without the need to burn a new S3 AMI image every time the system is brought down.

## Application and System Management

Common application management processes must be modified when running systems in AWS. Most system management tools were originally developed to control the hardware resources of an IT organization. These tools were later extended to manage software assets as well. Notwithstanding their ability to manage software assets, they use hardware management as their primary reference to organizing assets – that is, they usually organize assets based on the hardware they are running on.

Since Amazon offers no hardware access to end users, common system management tools are inappropriate for this environment and other resources must be used for system management.



# Key Cloud Computing Migration Issues

---

Amazon itself offers two different methods to manage AWS resources. The original tool is ElasticFox, a Firefox plug-in that lists and controls all AWS resources associated with a particular account.

A more recent tool that Amazon offers is an AJAX-enabled web page that may be accessed by any browser, not just Firefox. It too lists and controls all AWS resources for a given account.

Two primary challenges lie within the system management tools Amazon provides.

- Amazon tools manage Amazon resources, but require the user to separately manage the software assets running in EC2 instances. This means that two different management mechanisms must be used – one to manage the Amazon resources, and a second to manage the application software components running within the Amazon resources.
- The Amazon tools manage the Amazon resources as discreet instances; this means that in a three-tier application running on three or more EC2 instances the application cannot be managed as a whole, but each EC2 instance must be managed separately. This can pose problems for applications with large load variability; manual intervention is required to add or subtract resources to the application to enable it to handle varied load. Far more preferable would be a management system that would allow a unified approach to the application as a whole; should additional resources be required to meet increased demand, an instruction to increase application capability by a certain percentage could be implemented by the management tool, which would then instantiate whatever additional Amazon resources would be necessary to increase processing capacity by that percentage.

A number of third-parties have released tools to offer more sophisticated system management capabilities for AWS. Some of these tools manage aggregations of EC2 instances as a single application instance that can be scaled up or down, cloned to allow separate test and development application instances, and offer application versioning at the aggregated level. Other of the tools focus on managing the software components within individual EC2 instances. Amazon itself is likely to increase the breadth and functionality of its management tools in the near future.

Since the Lessonopoly application today runs on a single EC2 instance, for the initial implementation the current Amazon tools were selected for system management. This decision will be revisited in the future as traffic increases or the application is partitioned across multiple EC2 instances.



## Application and System Backup

Using redundant systems to prevent application unavailability due to hardware failure is useful; however, failing to have backups capable of restoring a system after database or hardware failure can literally make an application permanently unavailable. Consequently, reliable backups are critical to ensure long-term system availability.

Most organizations perform backups on machines in their data centers. They accomplish this either through manual activities like burning disks in machine DVD drives, or through automated solutions that copy system data out to disk- or tape-based repositories.

Since AWS offers no access to hardware, the manual approach is unworkable for EC2-based systems. However, automated backups are possible by using S3 as a reliable disk repository. Backups to S3 consist of new S3 AMI images of the core EC2 instance (in effect creating new “gold master” images if the core operating system or configuration files have changed) and separate S3 backup files of the dynamic data.

Because the Lessonopoly architecture uses EBS to store dynamic data, it is necessary to backup the EBS-based data separately from the EC2 instance backup, since S3-based instance backup does not capture data stored on EBS storage. In other words, AWS treats EBS as separate from the EC2 instance, and “burning” a new AMI “gold master” instance misses any EBS-based data. EBS storage instances must be separately backed up to S3, which forces a dual-backup strategy: one for the overall EC2 instance, and a second for the EBS instance attached to the EC2 image.

Fortunately, the EBS service offers the ability to create snapshots of the data in the EBS instance, which are then stored in S3. Subsequent snapshots store only changed data since the previous snapshot, thereby reducing overall storage requirements. Use of EBS snapshots as a dynamic data backup strategy is a significant improvement over direct backup of the EBS data into S3 files.

In its previous hosted arrangement, Lessonopoly backups were performed manually on a regular basis. In the AWS-hosted version, backups are performed with native AWS functionality, driven by cron scripts placed in the operating system running Lessonopoly. As the application topology complexity or data volumes increase, one of the third-party tools will be used to simplify the backup procedure.



## Lessonopoly AWS TCO

Calculating TCO for a cloud-based application is not necessarily a straightforward effort. Amazon pricing is public and easily understood, being based on units of resource consumption, like AMI instance use on a per-hour basis, Gigabyte of storage on a monthly basis, etc. However, it is often impossible to know before deployment exactly how much resource will be consumed; this will affect the total cost of the system on a monthly basis. Furthermore, since system use may be highly variable, which would affect total resource use, the monthly cost of an AWS application may be highly variable as well.

On the other hand, it is often difficult to evaluate the cost of an application that is hosted in-house or at a hosting provider. The monthly hosting fees for a single machine may be known, which makes evaluation easier; however, if a general hosting contract is in place, it may be difficult to establish the cost for a single server, or the financial elements of the contract may be handled by a different part of the organization, making it hard to discover the actual cost for hosting a single server.

Establishing the cost for an internally-hosted server (located in an organization's own data center) can be far more challenging. Most IT organizations track costs at a very high level, with no ability to assign costs at the a single server granularity level. Moreover, many of the costs associated with a server, like power or the imputed capital cost of the data center itself that should be assigned on a pro rata basis to a server, are not available at all.

As cloud computing becomes more common, the lack of transparency with respect to the established alternatives like internal and external hosting will become a significant issue. Most IT organizations will come under pressure to provide much more clarity regarding their true costs on a highly granular basis to enable TCO comparison among alternatives, including cloud computing.

For Lessonopoly, a TCO comparison was important because SVEF is a budget-constrained non-profit. Migrating the application to a cloud environment to improve system robustness was attractive, but if the cloud environment was not less than or equal to the equivalent cost of the hosted environment, SVEF would be forced to continue with the current hosting arrangement.



Fortunately, for the purposes of the Lessonopoly TCO comparison, the services offered by the hosting provider and Amazon are very similar: reliable machine support, with the ability to remotely power the machine on and off. The hosting provider offered no services beyond low-level machine support. All other support, like managing and patching the OS and ensuring the Lessonopoly application remained up and running, was left to SVEF. Therefore, the two environments – AWS and hosted – were very much alike and could be compared directly with respect to cost.

Because Lessonopoly is hosted on a single server today, but might expand to multi-tier or even multi-tier with one or more machines in each of the tiers, a number of TCO scenarios were evaluated. Each of the scenarios are described below, with an analysis of the cost for both alternatives.

### Scenario 1: Single Server

This is the current scenario, as displayed in Figure 5.

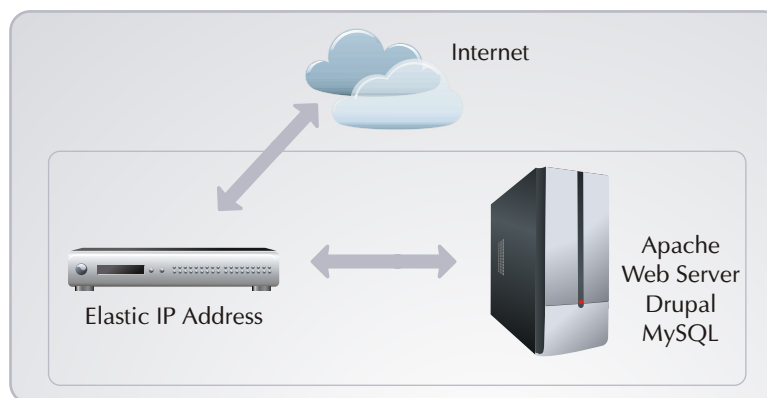


Figure 5

The costs of two alternatives can be seen in Table 1:

Hosted Cost		AWS Cost	
Server Monthly Hosting Fee	\$200.00	AMI Monthly Hosting Fee (Running 24 hours each day of month) <sup>1</sup>	\$74.00
Monthly Data Storage Fee (Included in Base Fee)	\$0.00	Monthly Data Storage Fee (50GB)	\$7.50
Monthly Data Transfer Fee (Included in Base Fee)	\$0.00	Monthly Data Transfer Fee (20GB)	\$3.50
		Monthly EBS Storage (50GB)	\$5.00
Total Monthly Cost	\$200.00	Total Monthly Cost	\$90.00
		Total % AWS Savings	55.00%



## Scenario 2: Multi-tier

If total traffic to the Lessonopoly website increases to the point at which performance of the system suffers due to load, the application could be partitioned to two machines, one hosting the Lessonopoly web server and Drupal instance, with the second hosting the Drupal database.

This scenario is displayed in Figure 6.

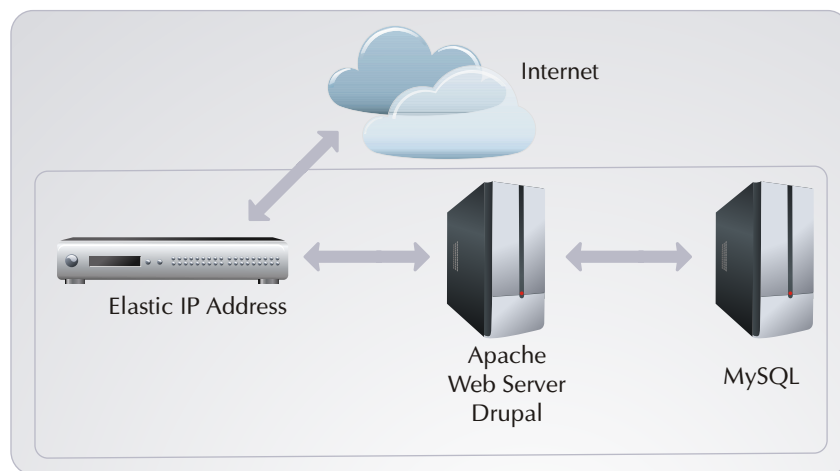


Figure 6

The costs of two alternatives can be seen in Table 2:

Hosted Cost		AWS Cost	
Server Monthly Hosting Fee (Two servers)	\$400.00	AMI Monthly Hosting Fee (Two servers running 24 hours each day of month) <sup>1</sup>	\$148.00
Monthly Data Storage Fee (Included in Base Fee)	\$0.00	Monthly Data Storage Fee (50GB X 2)	\$15.00
Monthly Data Transfer Fee (Included in Base Fee)	\$0.00	Monthly Data Transfer Fee (20GB X 2)	\$19.00
Amortized Monthly Cost of New Hardware <sup>1</sup>	\$55.00	Monthly EBS Storage (50GB X 2)	\$10.00
<b>Total Monthly Cost</b>	<b>\$455.00</b>	<b>Total Monthly Cost</b>	<b>\$192.00</b>
		<b>Total % AWS Savings</b>	<b>57.00%</b>



### Scenario 3: Multi-machine, multi-tier

If total traffic to the Lessonopoly website increases beyond the point at which partitioning of the two tiers onto individual machines is sufficient to support total application load, the application could be further partitioned to more machines, with two or more hosting the Lessonopoly web server and Drupal instance, and another machine hosting the Drupal database. To assess the TCO of this scenario, an application topology containing three web server machines was evaluated. In an application topology with multiple web server machines, a load balancer to distribute traffic among the web servers is necessary. Both hardware- and software-based load balancers are available in the market; since placing a hardware load balancer in the hosted environment would be difficult under the terms of the SVEF hosting agreement, and impossible in the AWS environment, this scenario assumes use of a software load balancing application. Because several open source load balancing packages are available, no software acquisition cost was factored into the evaluation. The load balancing software is run on a normal server; therefore, one additional server is placed into the scenario to host the load balancing software.

In summary, this scenario topology is:

- One server to host load balancing software
- Three servers to host Lessonopoly web servers
- One server to host Drupal database server

This scenario is displayed in Figure 7.

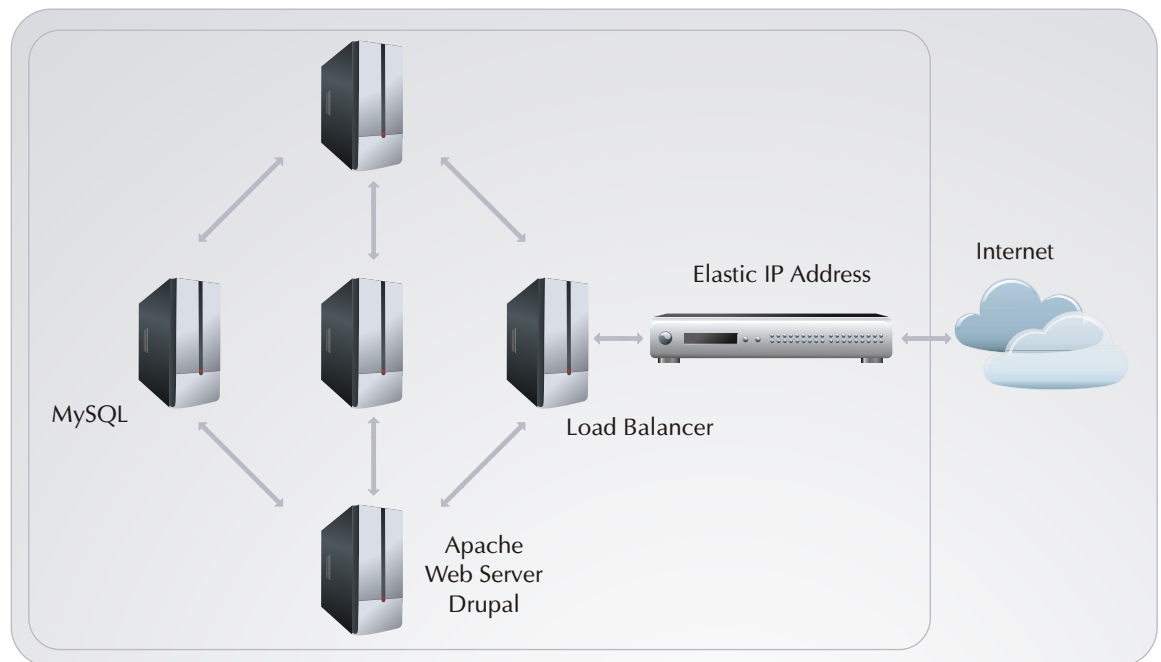


Figure 7



The costs of two alternatives can be seen in Table 3:

Hosted Cost		AWS Cost	
Server Monthly Hosting Fee (Five servers)	\$1,000.00	AMI Monthly Hosting Fee (Five servers running 24 hours each day of month) <sup>1</sup>	\$370.00
Monthly Data Storage Fee (Included in Base Fee)	\$0.00	Monthly Data Storage Fee (1TB)	\$150.00
Monthly Data Transfer Fee (Included in Base Fee)	\$0.00	Monthly Data Transfer Fee (1.7TB)	\$229.00
Amortized Monthly Cost of New Hardware <sup>1</sup>	\$222.00	Monthly EBS Storage (50GB X 5)	\$25.00
<b>Total Monthly Cost</b>	<b>\$1,222.00</b>	<b>Total Monthly Cost</b>	<b>\$774.00</b>
		<b>Total % AWS Savings</b>	<b>36.00%</b>

1. Note that Amazon is planning to offer load balancing as an AWS service in late 2009, which would obviate the need for one AMI instance

2. Note that in addition to the monthly hosting fee, four additional machines would need to be purchased to be placed in the hosting facility. Each machine would cost approximately \$2000. Amortized over an expected three year lifespan, the monthly cost would be \$222.

## Lessonopoly TCO Analysis

In each scenario, hosting the application in AWS offers savings, which range from 36% to 57%. Note that Scenario #3, which is the multi-machine, multi-tier topology, assigns an extra EC2 instance to host the load balancing software.

When Amazon releases its load balancing service in late 2009, this instance could be dropped and replaced with the Amazon service, which would reduce the cost of the AWS by \$74 per month, and raise the AWS percentage savings for this scenario to 38%. Of course, Amazon will charge for its load balancing service, so the savings would be somewhat less, but nevertheless, one could expect the percentage savings to be higher with substitution of the Amazon service for the load balancing software EC2 instance.

## Factors affecting TCO

The fees for the hosting provider as well as the costs for AWS EC2 instances are fixed; the cost for the hosting provider is a flat monthly fee, while the AWS EC2 instances are linear on a per-hour basis – running an EC2 instance every hour of the day for a full month sums to \$74.



Also note that Amazon Web Services offers what are called “Reserved Instances,” which, in return for an up-front payment, offer a lower per-hour cost. If an EC2 instance runs 24 hours per day, year-round, using Reserved Instances can reduce overall EC2 costs by approximately one-third. Reserved Instance costs were not factored into this TCO analysis, since this analysis is based on a monthly cost perspective; however, in a real-world situation that assumes an always-on system, Reserved Instances would undoubtedly be used to reduce the overall cost of Amazon Web Services computing.

However, the amount of storage used by the application and network traffic between users and the data center where the application is hosted can affect the total monthly cost. In this TCO analysis, we have not assigned any charge for these factors to the existing hosting arrangement because the monthly fee includes a certain amount of network traffic, and storage is based on what is available on the SVEF-owned machine located in the hosting provider’s data center.

For Amazon, S3 storage is priced on a sliding scale, according to volume, with the lowest volume tier costing \$.15/GB. EBS storage costs \$0.10/GB as well as \$0.10 per 1 million I/O requests in and out of an EBS instance. Network traffic in and out of Amazon’s data center is also priced on a sliding scale. For network traffic in, cost is \$.10 per GB. For network traffic out, pricing at the lowest tier is \$.17/GB.

Scenario One represents current Lessonopoly use and so data storage and transfer volumes of 50 GB and 20 GB, respectively, were used in TCO calculations. The other two scenarios used forecasts of data storage and transfer, based on likely application traffic increases in the future. Note that the increased data transfer requirements would outstrip the amounts included in the current external hosting fee and would increase that fee beyond \$200. The exact amount of increase was unavailable during the TCO analysis and was therefore not factored into the analysis; however, the increased fee would result in increasing the savings available through hosting the application in AWS.

In addition to the financial savings available to SVEF through using AWS to host its Lessonopoly application, AWS also offers increased application robustness by protecting SVEF from hardware failure. Using an external hosting provider avoids the need for SVEF to host machines in its own facilities and also avoids the need to manage service providers that supply resources like power and Internet connectivity.

However, the hosting provider expects SVEF to take responsibility for its own hardware. Because of budget limitations, SVEF is unable to purchase the redundant hardware necessary to protect itself from hardware failure.

Consequently, continuing the hosting arrangement exposes SVEF to the potential of application unavailability, which would affect the opportunity for teachers to improve their use of lesson plans and peer learning through the use of Lessonopoly. SVEF is particularly concerned about system uptime because if Lessonopoly is unavailable to any extent, teachers may conclude the application is too unreliable to depend on, and therefore avoid the use of the application altogether.



With the opportunity to save money as well as increase application reliability, SVEF will migrate the Lessonopoly application to AWS permanently. While this represents a change in the system hosting arrangements, with the potential for further change in the future, should system load require moving to a multi-tier system topology, end users are unaffected by the migration. Lessonopoly itself will continue to function in exactly the same fashion, with no discernible difference in user experience.

## Conclusion

The SVEF initiative to move its Lessonopoly application from a colocated hosting service to Amazon Web Services illustrates much of the allure of cloud computing. By leveraging cloud computing, SVEF was able to:

- Increase robustness: In the original hosting arrangement, the Lessonopoly application resided on a single hardware server. Even though good practices were followed regarding configuration management, source code management, and data backup, the application was still vulnerable to extended outages due to hardware failure. By migrating the application to AWS, SVEF mitigated its hardware vulnerability and increased the robustness of its application.
- Increase flexibility: If system load increased in the original hosting environment, little could be done in a short timeframe to increase system resources. By migrating the application to AWS, additional system resources can be made available almost immediately. Moreover, if system load then diminishes, it is easy to release the additional resources, with no further operational or financial commitment.
- Reduce costs: Even though the original hosting environment was relatively inexpensive, moving the Lessonopoly application reduced cost significantly. As the total application user base grows with additional traffic, the AWS infrastructure allows additional system resources to be deployed with linear scaling of costs. By contrast, if the original hosting infrastructure needed to have additional resources applied to the application, significant capital investment would be necessary, along with increasing hosting fees for each new piece of hardware.

While cloud computing may not be appropriate for every application or every computing environment, SVEF's experience with migrating Lessonopoly to AWS illustrates the very real benefits available to organizations that identify applications that are appropriate candidates for placement in cloud environments.



A HyperStratus White Paper

## **Migrating Applications to the Cloud**

An Amazon Web Services Case Study



# HyperStratus

Virtualization, Cloud Computing ... and more

### **About HyperStratus**

HyperStratus is a Silicon Valley-based cloud computing consultancy. Founded by executives with deep experience in corporate IT, enterprise software, global consultancies, and venture capital, HyperStratus helps its clients with a range of cloud computing services: strategy, cloud application architecture, technology selection, and system implementation. More information about HyperStratus may be found at [www.hyperstratus.com](http://www.hyperstratus.com).

[www.hyperstratus.com](http://www.hyperstratus.com)