



&&



Migrating your Existing Applications to the AWS Cloud

A Phase-driven Approach to Cloud Migration

Srishti Kakkar

srishti@amazon.com

February 2020

Abstract

With Amazon Web Services (AWS), you can provision compute power, storage and other resources, gaining access to a suite of elastic IT infrastructure services as your business demands them. With minimal cost and effort, you can move your application to the AWS cloud and reduce capital expenses, minimize support and administrative costs, and retain the performance, security, and reliability requirements your business demands.

This report will help you build a migration strategy for your company. It discusses steps, techniques and methodologies for moving your existing enterprise applications to the AWS cloud. There are several strategies for migrating applications to new environments. Through this report, we shall share several such strategies that will definitely help your company take advantage of the cloud. We discuss a phase-driven step-by-step strategy for migrating applications to the cloud.

More and more enterprises are moving applications to the cloud to modernize their current IT asset base or to prepare for future needs. They are taking the plunge, picking up a few mission-critical applications to move to the cloud and quickly realizing that there are other applications that are also a good fit for the cloud.

To illustrate the step-by-step strategy, we will discuss the motivation for the migration, describe the before and after application architecture, detail the migration process, and summarize the technical benefits of migration.

Introduction

Developers and architects looking to build *new applications* in the cloud can simply design the components, processes and workflow for their solution, employ the APIs of the cloud of their choice, and leverage the latest cloud-based best practices for design, development, testing and deployment. In choosing to deploy their solutions in a cloud-based infrastructure like Amazon Web Services (AWS), they can take immediate advantage of instant scalability and elasticity, isolated processes, reduced operational effort, on-demand provisioning and automation.

At the same time, many businesses are looking for better ways to migrate their *existing applications* to a cloud-based infrastructure so that they, too, can enjoy the same advantages seen with *greenfield* application development.

One of the key differentiators of AWS' infrastructure services is its flexibility. It gives businesses the freedom of choice to choose the programming models, languages, operating systems and databases they are already using or familiar with. As a result, many organizations are moving existing applications to the cloud today.

It is true that some applications ("IT assets") currently deployed in company data centers or co-located facilities might not make technical or business sense to move to the cloud or at least not yet. Those assets can continue to stay in place. However, we strongly believe that there are several assets within an organization that can be moved to the cloud today with minimal effort. This report will help **YZBuyer** build an enterprise application migration strategy for the organization. The step by step, phase-driven approach, discussed in this report will help you identify ideal projects for migration, build the necessary support within the organization and migrate applications with confidence.

Many organizations are taking incremental approach to cloud migration. It is very important to understand that with any migration, whether related to the cloud or not, there are one-time costs involved as well as resistance to change among the staff members (cultural and socio-political impedance). While these costs and factors are outside the scope of this report, you are advised to take into consideration these issues. Begin by building organizational support by evangelizing and training. Focus on long-term ROI as well as tangible and intangible factors of moving to the cloud and be aware of the latest developments in the cloud so that you can take full advantage of the cloud benefits.

There is no doubt that deploying your applications in the AWS cloud can lower your infrastructure costs, increases business agility and remove the undifferentiated "heavy lifting" within the enterprise. A successful migration largely depends on three things: the complexity of the application architecture; how loosely coupled your application is; and how much effort you are willing to put into migration.

A Phased Strategy for Migration: Step By Step Guide

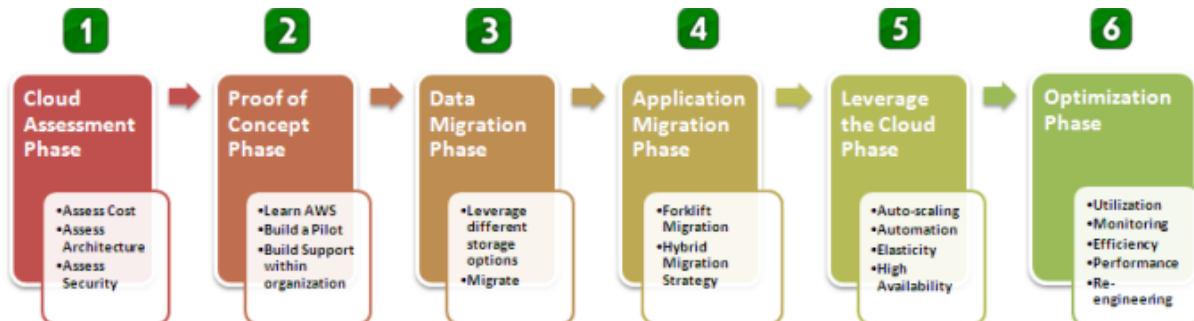


Figure 1: The Phase Driven Approach to Cloud Migration

Phase 1: Cloud Assessment Phase

AWS Pricing Philosophy

While the number and types of services offered by AWS has increased dramatically, our philosophy on pricing has not changed: you pay only for the resources that you use. The key tenets of the AWS pricing philosophy are:

- **Pay-as-you-go.** No minimum commitments or long-term contracts required. You replace your upfront capital expense with low variable cost and pay only for what you use. There is no need to pay upfront for excess capacity or get penalized for under-planning. For compute resources, you pay on an hourly basis from the time you launch a resource until the time you terminate it. For data storage and transfer, you pay on a per gigabyte basis. We charge based on the underlying infrastructure and services that you consume. You can turn off your cloud resources and stop paying for them when you don't need them.
- **Pay less when you reserve.** For certain products, you can invest in reserved capacity. In that case, you pay a low upfront fee and get a significantly discounted hourly rate, which results in overall savings between 42% and 76% (depending on the type of instance and the region in which you reserve) over equivalent on-demand capacity.
- **Pay even less per unit by using more.** You save more as you grow bigger. For storage and data transfer, pricing is tiered. The more you use, the less you pay per gigabyte. For compute, you get volume discounts up to 20% when you reserve more.
- **Pay even less as AWS grows.** Most importantly, we constantly focus on reducing our data center hardware costs, improving our operational efficiencies, lowering our power consumption, and generally lowering the cost of doing business. These optimizations and AWS's substantial and growing economies of scale result in passing savings back to you in the form of lower pricing. In the past six years, AWS has lowered pricing on 20 different occasions.

Leveraging Reserved Pricing in Total Cost of (Non) Ownership (TCO) Comparisons

Amazon Elastic Compute Cloud (Amazon EC2) and Amazon Relational Database Service (Amazon RDS) provide different ways to purchase an instance (virtual server) in the cloud. The On-Demand Instance pricing option lets you purchase an instance by the hour with no long-term commitments—you turn capacity on and off instantly. The Reserved Instance (RI) pricing option lets you make a low, one-time payment for each instance you want to reserve, and in turn, you receive a significant discount on the hourly usage charge for that instance (or in the case of the AWS GovCloud region's fixed price RI, no hourly fee at all), and are guaranteed capacity. The Spot Instance pricing option (available only for Amazon EC2) allows you to bid for unused compute capacity. Instances are charged at the Spot Price, which fluctuates periodically depending on the supply and demand for Spot Instance capacity. Functionally, Reserved Instances, OnDemand Instances, and Spot Instances are the same.

When you are comparing TCO, we highly recommend that you use the Reserved Instance (RI) pricing option in your calculations. They will provide the best apples-to-apples TCO comparison between on-premises and cloud infrastructure. Reserved Instances are similar to on-premises servers because in both cases, there is a one-time upfront cost. However, unlike on-premises servers, Reserved Instances can be “purchased” and provisioned within minutes—and you have the flexibility to turn them off when you don't need them and stop paying the hourly rate.

If you know how much you plan to utilize your Reserved Instances, you can save even more. AWS generally offers Light, Medium, and Heavy Utilization Reserved Instances. The Light Utilization model is a great option if you have periodic workloads that run only a couple of hours a day or a few days a week. Medium Utilization Reserved Instances are the same Reserved Instances that Amazon EC2 has offered for last several years. They are a great option if you don't plan to run your instances all the time, and if you want the option to shut down your instances when you're not using them. If you need a consistent baseline of capacity or if you run steady-state workloads, the Heavy Utilization model is the best option. In the AWS GovCloud region, full pre-pay for instances is available, which provides the deepest possible discount. Table 1 shows how much you can potentially save compared to running On-Demand Instances.

Reserved Instance Offering	Saving over On-Demand Instances	
Light Utilization Reserved Instances	Upto 42% (1 year)	Upto 56% (3 year)
Medium Utilization Reserved Instances	Upto 49% (1 year)	Upto 66% (3 year)
Heavy Utilization Reserved Instances	Upto 54% (1 year)	Upto 71% (3 year)
Fixed Price Reserved Instances	Upto 61% (1 year)	Upto 76% (3 year)

Table 1: Savings of Reserved Instance Types over On-Demand

Security and Compliance Assessment

If your organization has specific IT security policies and compliance requirements, we recommend that you involve your security advisers and auditors early in the process. At this stage, you can ask the following questions:

- What is my overall risk tolerance? Are there various classifications of my data that result in higher or lower tolerance to exposure?

- What are my main concerns around confidentiality, integrity, availability, and durability of my data?
- What are my regulatory or contractual obligations to store data in specific jurisdictions?
- What are my security threats? What is a likelihood of those threats materializing into actual attacks?
- Am I concerned about intellectual property protection and legal issues of my application and data?
- What are my options if I decide that I need to retrieve all of my data back from the cloud?
- Are there internal organizational issues to address to increase our comfort level with using shared infrastructure services?

Data security can be a daunting issue if not properly understood and analyzed. Hence, it is important that you understand your risks, threats (and likelihood of those threats), and then based on sensitivity of your data, classify the data assets into different categories (discussed in the next section). This will help you identify which datasets (or databases) to move to the cloud and which ones to keep in-house. It is also important to understand these important basics regarding AWS Security:

- You own the data, not AWS.
- You choose which geographic location to store the data. It doesn't move unless you decide to move it.
- You can download or delete your data whenever you like.
- You should consider the sensitivity of your data, and decide if and how you will encrypt your data while it is in transit and while it is at rest.
- You can set highly granular permissions to manage access of a user within your organization to specific service operations, data, and resources in the cloud for greater security control.

YZ Buyer Web Application Scenario: What will it look like when successfully migrated to AWS

As YZ Buyer wants to deploy its organization website—the official public-facing site that it uses to interact with prospects, customers, and partners. The website showcases all of the various brands of your agency and its subsidiaries, provides a listing of all the products and their specifications in an online catalog, lists all of the key stakeholders and the board of directors, and offers investor and public relations services.

The website attracts hundreds of thousands of visitors every month and is regularly accessed by thousands of customers outside the France. For the most part, the traffic flow is fairly steady state with small occasional blips every few months.

The website is a three-tier web application that leverages open source content management and publishing software, stores and serves a large amount of static media content (videos and PDFs) through a content delivery network, and uses a relational database to drive dynamic content that provides a personalized and interactive user experience. To support this website, let's assume the following compute resources:

- Two Linux servers for the web servers
- Two Linux servers for the application servers
- Two Linux servers for the MySQL database servers

Usage Graph The usage graph in Figure 2 shows an example traffic pattern for a steady-state web application. In order to meet this demand in the on-premises environment, you would order, pay for, install, and configure 6 physical servers. With AWS, you would have multiple options from which to choose.

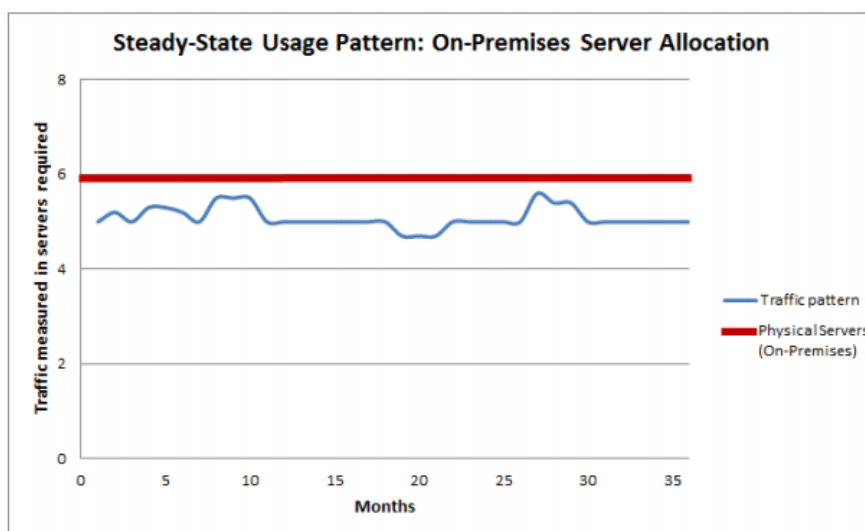


Figure 2: On-Premises Server Allocation for Steady-State Usage Pattern

TCO Comparison Across Options Considered

Table 2 compares the TCO of various AWS options vs. the on-premises alternative:

TCO	Web Application – Steady-State Usage Pattern			
Amortized Monthly Costs Over 3 Years	On-Premises Option	AWS Option 1 All Reserved (3-Year Heavy)	AWS Option 2 Mix of On-Demand and Reserved	AWS Option 3 All On-Demand
Compute/Server Costs				
Server Hardware	\$306	\$0	\$0	\$0
Network Hardware	\$62	\$0	\$0	\$0
Hardware Maintenance	\$47	\$0	\$0	\$0
Power and Cooling	\$172	\$0	\$0	\$0
Data Center Space	\$144	\$0	\$0	\$0
Personnel	\$1,200	\$0	\$0	\$0
AWS Instances	\$0	\$618	\$1,079	\$2,138
Total – Per Month	\$1,932	\$618	\$1,079	\$2,138
Total – 3 Years	\$69,552	\$22,260	\$38,859	\$76,982
Savings over On-Premises Option		68%	44%	-11%

Table 2: TCO Comparison – Steady-State Usage Pattern



**Recommended option
(most cost-effective)**

AWS Option 1: All Amazon EC2 Reserved Instances (3-Year Heavy Utilization)

In this option, we assume that you will purchase Reserved Instances for a 3-year term. Since this is a steady-state workload and you are planning to run these instances 24 hours per day, Heavy Utilization Reserved Instances is an attractive, cost-effective option.

Total monthly cost of 6 Reserved Instances amortized over a 3-year period:

2 web servers and 2 application servers: The instance type used is a High-Memory Extra Large, 3-Year Heavy Utilization Reserved Amazon EC2 Instance running in the EU Region at a rate of \$0.07 per hour with a onetime upfront cost of \$1,550. The amortized monthly cost for these servers is \$374.

2 database servers: The DB instance type used is a High-Memory Extra Large, 3-Year Reserved Amazon RDS DB Instance running in the EU Region with Master-Slave (Multi-AZ) configuration at a rate of \$0.011 per hour with a one-time upfront cost of \$1,550. The amortized monthly cost for these servers is \$244.

The total cost of running the steady-state web application (compute and database) on Reserved Instances for 3 years = \$22,260 (\$618 per month).

Summary

This is the most cost-effective option. You save 68% over the on-premises alternative. By purchasing 3-Year Heavy Utilization Reserved Instances, you get the maximum savings and lowest rates for your Amazon EC2 Instances and Amazon RDS DB Instances.

AWS Option 2: Mix of Amazon EC2 Reserved Instances (3-Year Heavy Utilization) and On-Demand Instances

In this option, we assume you will purchase 3-year Heavy Utilization Reserved Instances for the minimum number of servers you need to run your application (i.e. your baseline), thereby reducing your total upfront commitment. For additional servers, we assume you will leverage On-Demand Instances.

Please note that you can purchase Reserved Instances anytime. Unlike the on-premises option with Reserved Instances, you don't have to plan ahead for capacity or allocate the time it takes to build out physical data center capacity. When you purchase Reserved Instances, your billing will automatically switch from the On-Demand Instance hourly rate to the Reserved Instance discounted hourly rate.

Baseline (minimum servers needed to run a three-tier web application)

Total monthly cost of 4 Reserved Instances amortized over a 3-year period:

1 web server and 1 application server: The instance type used is a High-Memory Extra Large, 3-Year Heavy Utilization Reserved Amazon EC2 Instance running in the EU Region at a rate of \$0.07 per hour with a one-time upfront cost of \$1,550. The amortized monthly cost for these servers is \$187.

2 database servers: The DB instance type used is a High-Memory, Extra Large 3-Year Heavy Utilization Reserved Amazon RDS DB Instance running in the EUt Region with Master-Slave (Multi-AZ) configuration at a rate of \$0.011 per hour, with a one-time upfront cost of \$1,550. The amortized monthly cost for these servers is \$244.

Peak (additional servers needed)

Total monthly cost of 2 On-Demand Instances amortized over a 3-year period:

1 web server and 1 application server: The instance type used is a High-Memory Extra Large, On-Demand Amazon EC2 Instance running in the EU Region at a rate of \$0.45 per hour for 24 hours/day (Always-on). The amortized monthly cost for these servers is \$648.

The total cost of running the steady-state web application (compute and database) on Reserved Instances for 3 years = \$38,859 (\$1,079 per month).

Summary

This option offers 44% savings over the on-premises alternative. It's a lower upfront commitment (\$6,200) than either AWS Option 1 (\$9,300) or the on-premises option (\$14,952). If you're not confident about your peak capacity needs or if you want to have a little more flexibility while still

saving costs, you might choose this option. However, since this is a steady-state workload where demand is largely predictable, we still recommend the AWS Option 1 over this more flexible AWS Option 2.

AWS Option 3: All Amazon EC2 On-Demand Instances

In this option, we assume that you will choose On-Demand Instances to run your steady-state web application. Unlike the on-premises option, with On-Demand Instances, you don't have to plan ahead for capacity or purchase any resources ahead of time. You simply start and stop your Amazon EC2 Instances and Amazon RDS DB Instances for the hours you want to use. You are billed every month based on your usage. In this case, since this is a steady-state workload, we assume you will keep the instances running for 24 hours per day.

Total monthly cost of 6 On-Demand Instances:

4 web and application servers: The instance type used is a High-Memory Extra Large, On-Demand Amazon EC2 Instance running in the EU Region at a rate of \$0.45 per hour for 24 hours/day (Always-on).

2 database servers: The DB instance type used is a High-Memory Extra Large, On-Demand Amazon RDS DB Instance running in the EU Region at a rate of \$0.585 per hour for 24 hours/day (Always-on).

The total cost of running the steady-state web application (compute and database) on On-Demand Instances for 3 years = \$76,982 (\$2,138 per month).

Summary

With AWS, you have an option to choose zero upfront commitment and leverage On-Demand Instances for your steady-state workloads. Some AWS customers prefer this option because it allows them to start small with no upfront commitment, and provides maximum flexibility while reducing risk to close to zero. For only a 11% cost premium over on-premises infrastructure—which requires 100% upfront purchase and very little flexibility—they have an environment that can be started up or completely shut down to zero at a moment's notice. And, of course, you can always optimize your cost later by replacing On Demand instances with Reserved Instances

Recommended Option for Steady-State Web Application: 3-Year Heavy Utilization Reserved Instances

As you can see from the above calculations, if you have a web application with uniform steady-state traffic, the most cost-effective option is to use 3-Year Heavy Utilization Reserved Instances (AWS Option 1). This option offers 68% savings over the on-premises alternative.

Phase 2: Proof of Concept Phase

Once you have identified the right candidate for the cloud and estimated the efforts required to migrate, it's time to test the waters with a small proof of concept. The goal of this phase is to learn AWS and ensure that your assumptions regarding suitability for migration to the cloud are accurate. In this phase, you can deploy application and, in the process, begin to get your feet wet with the AWS cloud.

Get your feet wet with AWS

Get familiar with the AWS API, AWS tools, SDKs, Firefox plug-ins and most importantly the AWS Management Console and command line tools. At a minimum, at the end of this stage, you should know how to use the AWS Management Console (or the Firefox plugins) and command line tools to do the following:

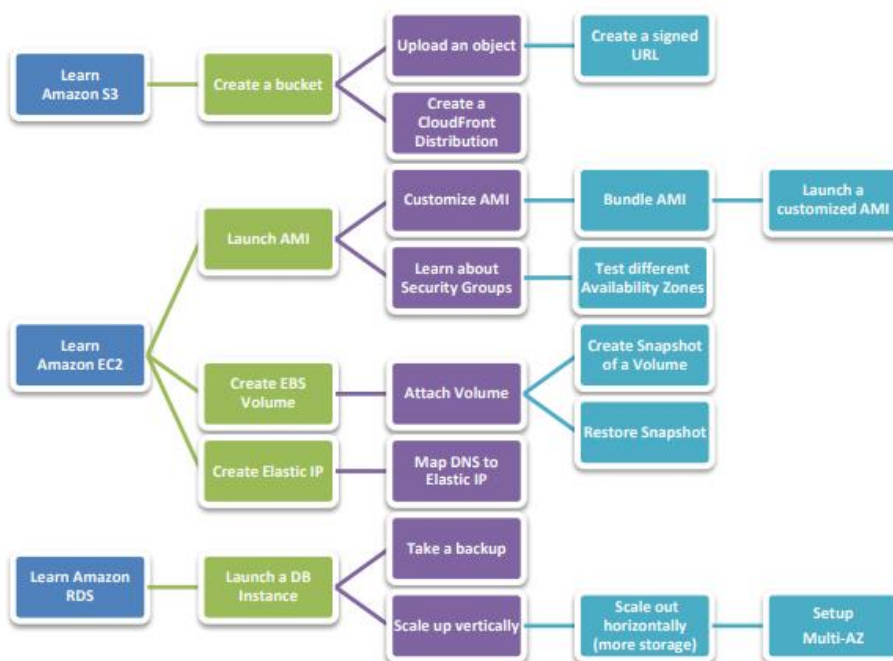


Figure 3: Minimum items to learn about services in a Proof of Concept

Learn about the AWS security features

Be aware of the AWS security features available today. Use them at every stage of the migration process as you see fit. During the Proof of Concept Phase, learn about the various security features provided by AWS: AWS credentials, Multi Factor Authentication (MFA), authentication and authorization. At a minimum, learn about the AWS Identity and Access Management (IAM) features that allow you to create multiple users and manage the permissions for each of these users within your AWS Account. Figure 4 highlights the topics you need to learn regarding IAM:

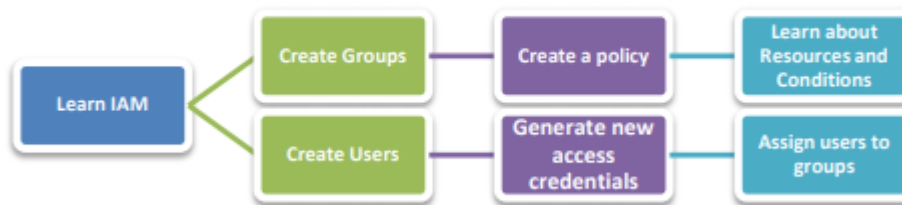


Figure 5: Minimum items to learn about security in a Proof of Concept Phase

At this stage, you want to start thinking about whether you want to create different IAM groups for different business functions within your organization or create groups for different IT roles (admins, developers, testers etc.) and whether you want to create users to match your organization chart or create users for each application.

Build a Proof-Of-Concept Build a proof-of-concept: represents a microcosm of your application, or which tests critical functionality of your application in the cloud environment. Start with a small database (or a dataset); don't be afraid of launching and terminating instances, or stress-testing the system.

As you are thinking of migrating a web application, you can start by deploying miniature models of all the pieces of your architecture (database, web application, load balancer) with minimal data. In the process, learn how to build a Web Server AML, how to set the security group so that only the web server can talk to the app server, how to store all the static files on Amazon S3 and mount an EBS volume to the Amazon EC2 instance, how to manage/monitor your application using Amazon CloudWatch and how to use IAM to restrict access to only the services and resources required for your application to function.

Most of our enterprise customers dive into this stage and reap tremendous value from building pilots. We have noticed that customers learn a lot about the capabilities and applicability of AWS during the process and quickly broaden the set of applications that could be migrated into the AWS cloud.

In this stage, you can build support in your organization, validate the technology, test legacy software in the cloud, perform necessary benchmarks and set expectations.

At the end of this phase, you should be able to answer the following questions:

- Did I learn the basic AWS terminology (instances, AMLs, volumes, snapshots, distributions, domains and so on)?
- Did I learn about many different aspects of the AWS cloud (compute, storage, network, database, security) by building this proof of concept?
- Will this proof of concept support and create awareness of the power of the AWS cloud within the organization?
- What is the best way to capture all the lessons that I learned? A whitepaper or internal presentation?
- How much effort is required to roll this proof-of-concept out to production?

- Which applications can I immediately move after this proof of concept?

After this stage, you will have far better visibility into what is available with AWS today. You will get hands-on experience with the new environment which will give you more insight into what hurdles need to be overcome in order to move ahead.

YZ Buyer Web Application Scenario: Proof of Concept when achieved

Proof of Concept

The web development team was skeptical about the relational database migration. To test, they decided to build a proof of concept application. During the proof of concept, **the team learned the following techniques: starting, terminating and configuring Amazon EC2 instances and Amazon RDS DB Instances, storing and retrieving Amazon S3 objects, and setting up elastic load balancers using the AWS Management Console.** They learned a ton about AWS and saw that they have full control over the environment, and felt a lot more confident about moving to the next step. The relational database files (binary and transaction logs) were moved to Amazon RDS instances using the standard “mysqlimport” utility. For the test environment, they deployed a DB Instance within a single Availability Zone, and for the production environment, they set up a DB Instance with the Multi-AZ deployment to increase availability. The team was able to successfully test and migrate all data to a DB instance, get performance metrics using Amazon CloudWatch, and set retention policies for backups. They built migration scripts to automate the process and created awareness within the organization by organizing a “brownbag” session and successfully demonstrated their work to their peers.

Phase 3: Data Migration Phase

In this phase, enterprise architects should ask following questions:

- What are the different storage options available in the cloud today?
- What are the different RDBMS (commercial and open source) options available in the cloud today?
- What is my data segmentation strategy? What trade-offs do I have to make?
- How much effort (in terms new development, one-off scripts) is required to migrate all my data to the cloud?

Understand Various Storage Options Available in the AWS Cloud

The table will help explain which storage option to use when:

	Amazon S3 + CloudFront	Amazon EC2 Ephemeral Store	Amazon EBS	Amazon SimpleDB	Amazon RDS
Ideal for	Storing large write-once, read-many types of objects, Static Content Distribution	Storing non-persistent transient updates	Off-instance persistent storage for any kind of data,	Query-able light-weight attribute data	Storing and querying structured relational and referential data
Ideal examples	Media files, audio, video, images, Backups, archives, versioning	Config data, scratch files, TempDB	Clusters, boot data, Log or data of commercial RDBMS like Oracle, DB2	Querying, Indexing Mapping, tagging, click-stream logs, metadata, Configuration, catalogs	Web apps, Complex transactional systems, inventory management and order fulfillment systems Clusters
Not recommended for	Querying, Searching	Storing database logs or backups, customer data	Static data, Web-facing content, key-value data	Complex joins or transactions, BLOBs Relational, Typed data	
Not recommended examples	Database, File Systems	Shared drives, Sensitive data	Content Distribution	OLTP, DW cube rollups	Clustered DB, Simple lookups

Table 3: Data Storage Options in AWS Cloud

Migrate your Fileserver systems, Backups and Tape Drives to Amazon S3

If your existing infrastructure consists of Fileservers, Log servers, Storage Area Networks (SANs) and systems that are backing up the data using tape drives on a periodic basis, you should consider storing this data in Amazon S3. Existing applications can utilize Amazon S3 without major change. If your system is generating data every day, the recommended migration flow is to point your “pipe” to Amazon S3 so that new data is stored in the cloud right away. Then, you can have an independent batch process to move old data to Amazon S3. Most enterprises take advantage of their existing encryption tools (256-bit AES for data at-rest, 128-bit SSL for data in-transit) to encrypt the data before storing it on Amazon S3.

Migrate your MySQL Databases to Amazon RDS

If you use a standard deployment of MySQL, moving to Amazon RDS will be a trivial task. Using all the standard tools, you will be able to move and restore all the data into an Amazon RDS DB instance. After you move the data to a DB instance, make sure you are monitoring all the metrics you need. It is also highly recommended that you set your retention period so AWS can automatically create periodic backups.

Migrate your Commercial Databases to Amazon EC2 using Relational DB AMIs

If you require transactional semantics (commit, rollback) and are running an OLAP system, simply use traditional migration tools available with Oracle, MS SQL Server, DB2 and Informix. All of the major databases are available as Amazon Machine Images and are supported in the cloud by the vendors. Migrating your data from an on-premise installation to an Amazon EC2 cloud instance is no different than migrating data from one machine to another.

Move Large Amounts of Data using Amazon Import/Export Service

When transferring data across the Internet becomes cost or time prohibitive, you may want to consider the AWS Import/Export service. With AWS Import/Export Service, you load your data on USB 2.0 or eSATA storage devices and ship them via a carrier to AWS. AWS then uploads the data into your designated buckets in Amazon S3.

For example, if you have multiple terabytes of log files that need to be analyzed, you can copy the files to a supported device and ship the device to AWS. AWS will restore all the log files in your designated bucket in Amazon S3, which can then be fetched by your cloud-hosted business intelligence application or Amazon Elastic MapReduce services for analysis.

If you have a 100TB Oracle database with 50GB of changes per day in your data center that you would like to migrate to AWS, you might consider taking a full backup of the database to disk then copying the backup to USB 2.0 devices and shipping them. Until you are ready to switch the production DBMS to AWS, you take differential backups. The full backup is restored by the import service and your incremental backups are transferred over the Internet and applied to the DB Instance in the cloud. Once the last incremental backup is applied, you can begin using the new database server.

Buyer Web Application Scenario: Data Migration when achieved

Once the proof of concept was complete, the team decided to **move all of the application's static files** (Images, JS, CSS, video, audio, and static HTML content) into an Amazon S3 bucket, created a CloudFront distribution of that Amazon S3 bucket, and modified the references in web pages so that end-users get the content directly from Amazon S3 and Amazon CloudFront. With a few scripts and the AWS SDK for Java library, they were able to **transfer all data from tape drives and upload it to Amazon S3**.

Phase 4: Application Migration Phase

Forklift Migration Strategy Stateless applications, tightly coupled applications, or self-contained applications might be better served by using the forklift approach. Rather than moving pieces of the system over time, forklift or “pick it all up at once” and move it to the cloud. Self-contained Web applications that can be treated as single components and backup/archival systems are examples of these types of systems that can be moved into the cloud using this strategy. Components of a 3-tier web application that require extremely-low latency connectivity between them to function and cannot afford internet latency might be best suited to this approach if the entire application including the web, app and database servers, is moved to the cloud all at once.

In this approach, you might be able to migrate an existing application into the cloud with few code changes. Most of the changes will involve copying your application binaries, creating and configuring Amazon Machine Images, setting up security groups and elastic IP addresses, DNS, switching to Amazon RDS databases. This is where AWS's raw infrastructure services (Amazon EC2, Amazon S3, Amazon RDS and Amazon VPC) really shine.

In this strategy, the applications might not be able to take immediate advantage of the elasticity and scalability of the cloud because, after all, you are swapping real physical servers with EC2 instances, or replacing file servers with Amazon S3 buckets or Amazon EBS volumes; logical components matter less than the physical assets. However, it's important to realize that, by using this approach for certain application types, you are shrinking your IT infrastructure footprint (one less thing to worry about) and offloading the undifferentiated heavy lifting to AWS. This enables you to focus your resources on things that actually differentiate you from your competitors. You will revisit this application in the next stages and will be able to realize even more benefits of the cloud.

Hybrid Migration Strategy A hybrid migration consists of taking some parts of an application and moving them to the cloud while leaving other parts of the application in place. The hybrid migration strategy can be a low-risk approach to migration of applications to the cloud. Rather than moving the entire application at once, parts can be moved and optimized one at a time. This reduces the risk of unexpected behavior after migration and is ideal for large systems that involve several applications. For example, if you have a website and several batch processing components (such as indexing and search) that power the website, you can consider using this approach. The batch processing system can be migrated to the cloud first while the website continues to stay in the traditional data center. The data ingestion layer can be made “cloud-aware” so that the data is directly fed to an Amazon EC2 instance of the batch processing system before every job run. After proper testing of the batch processing system, you can decide to move the website application.


Configuring and Creating your AMIs In many cases, it is best to begin with AMIs either provided by AWS or by a trusted solution provider as the basis of AMIs you intend to use going forward. Depending on your specific requirements, you may also need to leverage AMIs provided by other ISVs. In any case, the process of configuring and creating your AMIs is the same. It is recommended that you create an AMI for each component designed to run in a separate Amazon EC2 instance. It is also recommended to create an automated or semi-automated deployment process to reduce the time and effort for re-bundling AMIs when new code is released. This would be a good time to begin thinking about a process for configuration management to ensure your servers running in the cloud are included in your process.

Buyer Web Application Scenario: Application Migration when achieved

During the application migration phase, the development team launched both small and large instances for their web and tomcat servers. They created AMIs (Amazon Machine Images, basically “golden” system images) for each server type. AMIs were designed to boot directly from an EBS volume and fetch the latest WAR file binaries during launch from the source code repository. They modified their build and deployment scripts to use the cloud as an endpoint. Security Groups were defined to isolate web servers from the applications and database servers. Testing (functional, load, performance etc.) was performed to ensure that the systems were performing at expected levels, and that exit criteria for each component were met.

Co-existence Phase

During the migration phase, the collocation infrastructure was not deprecated immediately.

 **Buyer** employed a hybrid migration strategy during the migration of all web and application servers. The configuration of the onpremise hardware load balancer was modified to send requests to the new instances in the cloud. For a short duration, the load balancer was routing traffic to the servers in the cloud in as well as to the physical servers. After verifying that the servers in the cloud were performing at required levels, the physical servers were dismissed one by one, the load balancers were updated, and all of the web traffic was being served up by the EC2 instances running in the cloud.

After testing was completed, the DNS was switched to point to the cloud-based web servers and the application was fully migrated to the AWS cloud.

Phase 5: Leverage the Cloud

After you have migrated your application to the cloud, run the necessary tests, and confirmed that everything is working as expected, it is advisable to invest time and resources to determine how to leverage additional benefits of the cloud.

Leverage other AWS services

Auto Scaling Service

Auto Scaling enables you to set conditions for scaling up or down your Amazon EC2 usage. When one of the conditions is met, Auto Scaling automatically applies the action you've defined. Examine each cluster of similar instances in your Amazon EC2 fleet and see whether you can create an Auto Scaling group and identify the criteria of scaling automatically (CPU utilization, network I/O etc.)

At minimum, you can create an Auto Scaling group and set a condition that your Auto Scaling group will always contain a fixed number of instances. Auto Scaling evaluates the health of each Amazon EC2 instance in your Auto Scaling group and automatically replaces unhealthy Amazon EC2 instances to keep the size of your Auto Scaling group constant.

Amazon CloudFront

With just a few clicks or command line calls, you can create an Amazon CloudFront distribution for any of your Amazon S3 buckets. This will edge cache your static objects closer to the customer and reduce latency. This is often so easy to do that customers don't wait until this phase to take advantage of CloudFront; they do so much earlier in the plan.

Automate Elasticity

Elasticity is a fundamental property of the cloud. Elasticity can be implemented at different levels of the application architecture. Implementing elasticity might require refactoring and decomposing your application into components so that it is more scalable. The more you can automate elasticity in your application, the easier it will be to scale your application horizontally and therefore the benefit of running it in the cloud is increased.

In this phase, you should try to automate elasticity. After you have moved your application to AWS and ensured that it works, there are 3 ways to automate elasticity at the stack level. This enables you to quickly start any number of application instances when you need them and terminate them when you don't, while maintaining the application upgrade process. Choose the approach that best fits your software development lifestyle.

1. Maintain Inventory of AMIs It's easiest and fastest to setup inventory of AMIs of all the different configurations but difficult to maintain as newer versions of applications might mandate updating the AMIs.
2. Maintain a Golden AMI and fetch binaries on boot This is a slightly more relaxed approach where a base AMI ("Golden Image") is used across all application types across the organization while the rest of the stack is fetched and configured during boot time.
3. Maintain a Just-Enough-OS AMI and a library of recipes or install scripts. This approach is probably the easiest to maintain especially when you have a huge variety of application stacks to deploy. In this approach, you leverage the programmable infrastructure and maintain a library of install scripts that are executed on-demand.

YZ Buyer Web Application Scenario: Leveraging the Cloud

Once the production site was launched, YZBuyer was looking forward to the time when they could use some of the advanced features of AWS. The team automated some processes so that once the server is started it could be easily “attached” to the topology. They created an **Auto Scaling group** of web servers and were able to provision more capacity automatically when specific resources reach a certain threshold (Apache web servers CPU utilization above 80% for 10 min). The team invested some time and resources in streamlining their development and testing processes to make it is easy to clone testing environments. They gained lot of experience using AWS resources and also invested time in **leveraging multiple Availability Zones** for even higher availability.

Phase 6: Optimization Phase

Understanding your Usage Patterns With the cloud, you don’t have to master the art of capacity planning because you have the ability to create an automated elastic environment. If you can understand, monitor, examine and observe your load patterns, you can manage this elastic environment more effectively. You can be more proactive if you understand your traffic patterns. For example, if your customer-facing website, deployed in AWS global infrastructure, does not expect any traffic from certain part of the world in certain time of the day, you can scale down your infrastructure in that AWS region for that time. The closer you can align your traffic to cloud resources you consume, the higher the cost savings will be.

Terminate the Under-Utilized Instances Inspect the system logs and access logs periodically to understand the usage and lifecycle patterns of each Amazon EC2 instance. Terminate your idle instances. Try to see whether you can eliminate under-utilized instances to increase utilization of the overall system. For example, examine the application that is running on an m1.large instance (1X \$0.40/hour) and see whether you can scale out and distribute the load across to two m1.small instances (2 X \$0.10/hour) instead.

Leverage Amazon EC2 Reserved Instances Reserved Instances give you the option to make a low, one-time payment for each instance you want to reserve and in turn receive a significant discount on the hourly usage charge for that instance. When looking at usage patterns, try to identify instances that are running in steady-state such as a database server or domain controller. You may want to consider investing in Amazon EC2 Reserved Instances (3 year term) for servers running above 24% or higher utilization. This can save up to 49% of the hourly rate.

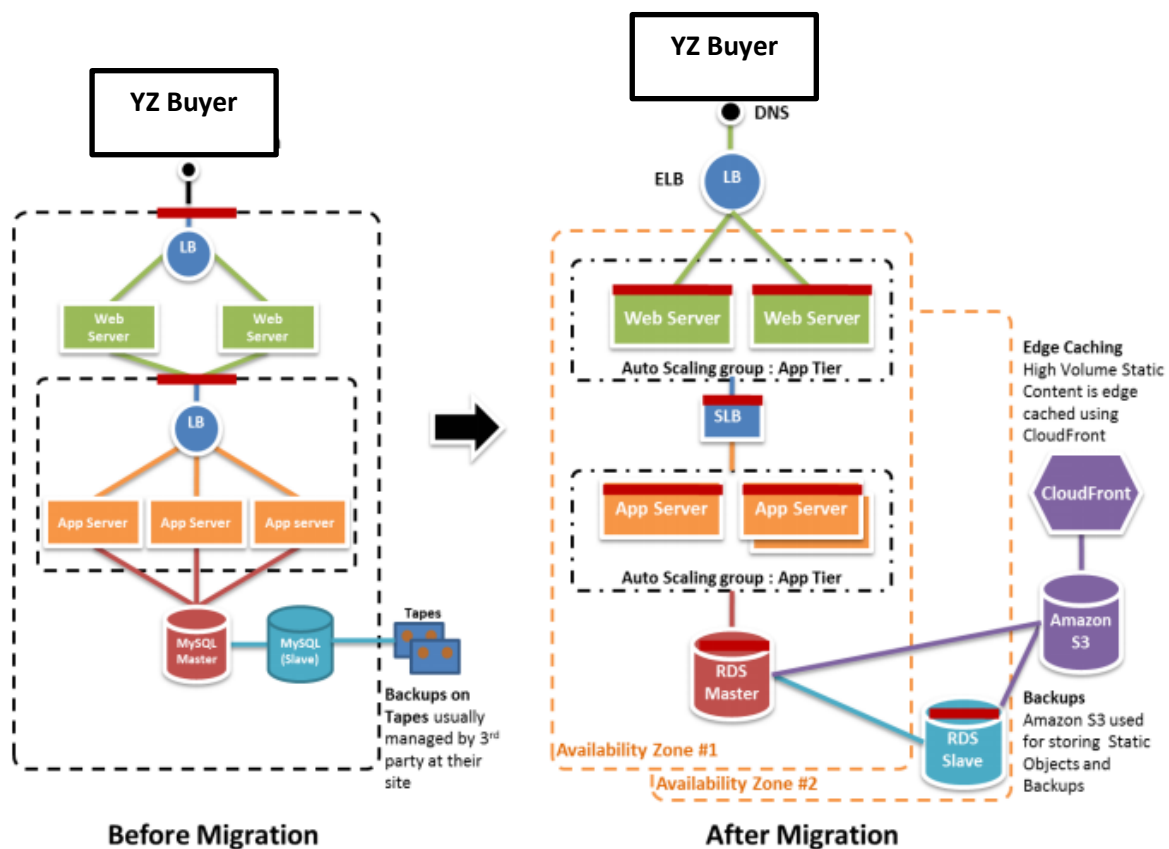
Track your AWS Usage and Logs Monitor your AWS usage bill, Service API usage reports, Amazon S3 or Amazon CloudFront access logs periodically.

Maintain Security of Your Applications Ensure that application software is consistent and always up to date and that you are patching your operating systems and applications with the latest vendor security updates. Patch an AMI, not an instance and redeploy often; ensure that the latest AMI is deployed across all your instances.

YZ Buyer Web Application Scenario: Optimization Phase

During the optimization phase, the development team analyzed their utilization patterns and realized that they could save 30% if they switched to Reserved Instances. They purchased four Reserved Instances (2 for web servers and the other 2 for tomcat servers). They built additional scripts to run their web application in 3 different “modes”: weekend, weekday and promotion mode. These modes defined the minimum number of servers to run. The team also integrated Amazon CloudWatch into their existing dashboards so that they can monitor the system metrics of every instance in their cloud fleet.

Conclusion



The company was able to successfully migrate an existing web application to the AWS cloud. With minimal effort, the team was not only able to free up the physical infrastructure for other projects but also reduce the operating expenditure by 30%. Using the phased-driven approach, the development team was able to resolve all the financial, technical and social-political concerns. Deciding to invest in a proof of concept proved extremely valuable. The resulting architecture was not only elastic and scalable but also flexible and easier to maintain.