



# Migrating your Existing Applications to the AWS Cloud

A Phase-driven Approach to Cloud Migration

*Jinesh Varia*  
*jvaria@amazon.com*

*October 2010*



## 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 paper helps you build a migration strategy for your company. It discusses steps, techniques and methodologies for moving your existing enterprise applications to the AWS cloud. To get the most from this paper, you should have basic understanding of the different products and features from Amazon Web Services.

There are several strategies for migrating applications to new environments. In this paper, we shall share several such strategies that help enterprise companies 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 provide three scenarios listed in the table. Each scenario discusses the motivation for the migration, describes the before and after application architecture, details the migration process, and summarizes the technical benefits of migration:

Scenario Name	Solution	Use case	Motivation For migration	Additional Benefits	Services Used
Company A	Web Application	Marketing and collaboration Web site	Scalability + Elasticity	Auto Scaling, pro-active event based scaling	EC2, S3, EBS, SimpleDB, AS, ELB, CW, RDS
Company B	Batch processing pipeline	Digital Asset Management Solution	Faster time to market	Automation and improved development productivity	EC2, EBS, S3, SQS
Company C	Backend processing workflow	Claims Processing System	Lower TCO, Redundancy	Business continuity and Overflow-protection	EC2, S3, EBS, AS, SQS, IE

## 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<sup>1</sup> 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 paper will help you build an enterprise application migration strategy for your organization. The step by step, phase-driven approach, discussed in the paper 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 technical paper, 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. We have noticed that when customers have followed the step by step approach (discussed in this paper) and have invested time and resources towards building proof of concept projects, they clearly see the tremendous potential of AWS, and are able to leverage its strengths very quickly.

---

<sup>1</sup> Architecting for the Cloud: Best Practices Whitepaper - [http://media.amazonwebservices.com/AWS\\_Cloud\\_Best\\_Practices.pdf](http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf)

# A Phased Strategy for Migration: Step By Step Guide

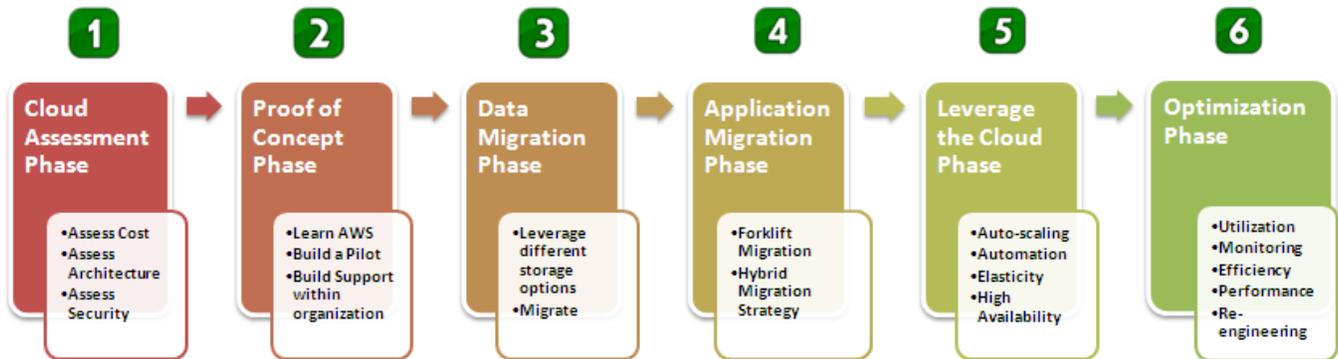


Figure 1: The Phase Driven Approach to Cloud Migration

Phases	Benefits
<b>Cloud Assessment</b> <ul style="list-style-type: none"> <li>Financial Assessment (TCO calculation)</li> <li>Security and Compliance Assessment</li> <li>Technical Assessment (Classify application types)</li> <li>Identify the tools that can be reused and the tools that need to be built</li> <li>Migrate licensed products</li> <li>Create a plan and measure success</li> </ul>	Business case for migration (Lower TCO, faster time to market, higher flexibility & agility, scalability + elasticity)  Identify gaps between your current traditional legacy architecture and next -generation cloud architecture
<b>Proof of Concept</b> <ul style="list-style-type: none"> <li>Get your feet wet with AWS</li> <li>Build a pilot and validate the technology</li> <li>Test existing software in the cloud</li> </ul>	Build confidence with various AWS services  Mitigate risk by validating critical pieces of your proposed architecture
<b>Moving your Data</b> <ul style="list-style-type: none"> <li>Understand different storage options in the AWS cloud</li> <li>Migrate file servers to Amazon S3</li> <li>Migrate commercial RDBMS to EC2 + EBS</li> <li>Migrate MySQL to Amazon RDS</li> </ul>	Redundancy, Durable Storage, Elastic Scalable Storage  Automated Management Backup
<b>Moving your Apps</b> <ul style="list-style-type: none"> <li>Forklift migration strategy</li> <li>Hybrid migration strategy</li> <li>Build “cloud-aware” layers of code as needed</li> <li>Create AMIs for each component</li> </ul>	Future-proof scaled-out service-oriented elastic architecture
<b>Leveraging the Cloud</b> <ul style="list-style-type: none"> <li>Leverage other AWS services</li> <li>Automate elasticity and SDLC</li> <li>Harden security</li> <li>Create dashboard to manage AWS resources</li> <li>Leverage multiple availability zones</li> </ul>	Reduction in CapEx in IT Flexibility and agility Automation and improved productivity Higher Availability (HA)
<b>Optimization</b> <ul style="list-style-type: none"> <li>Optimize usage based on demand</li> <li>Improve efficiency</li> <li>Implement advanced monitoring and telemetry</li> <li>Re-engineer your application</li> <li>Decompose your relational databases</li> </ul>	Increased utilization and transformational impact in OpEx  Better visibility through advanced monitoring and telemetry

The order of the phases is not important. For example, several companies prefer to skip Phase 1 (Assessment Phase) and dive right into Phase 2 (Proof of Concept) or perform Application Migration (Phase 4) before they migrate all their data (Phase 3).

## Phase 1: Cloud Assessment Phase

This phase will help you build a business case for moving to the cloud.

### Financial Assessment

Weighing the financial considerations of owning and operating a data center or co-located facilities versus employing a cloud-based infrastructure requires detailed and careful analysis. In practice, it is not as simple as measuring potential hardware expense alongside utility pricing for compute and storage resources. Indeed, businesses must take a multitude of options into consideration in order to affect a valid comparison between the two alternatives. Amazon has published a whitepaper, [The Economics of the AWS cloud<sup>2</sup>](#) to help you gather the necessary data for an appropriate comparison. This basic TCO methodology and the accompanying Amazon EC2 Cost Calculator uses industry data, AWS customer research, and user-defined inputs to compare the annual fully-burdened cost of owning, operating, and maintaining IT infrastructure with the pay-for-use costs of Amazon EC2. Note that this analysis compares only the direct costs of the IT infrastructure and ignores the many indirect economic benefits of cloud computing, including high availability, reliability, scalability, flexibility, reduced time-to-market, and many other cloud-oriented benefits. Decision makers are encouraged to conduct a separate analysis to quantify the economic value of these features.

Pricing Model	One-time Upfront			Monthly		
	AWS	Co-lo	On-Site	AWS	Co-lo	On-Site
Server Hardware	0	\$\$\$	\$\$	\$\$	0	0
Network Hardware	0	\$\$	\$\$	0	0	0
Hardware Maintenance	0	\$\$	\$\$	0	0	\$
Software OS	0	\$\$	\$\$	\$	0	0
Power and Cooling	0	0	\$\$	0	\$\$	\$
Data Center/Co-located Space	0	\$\$	\$\$	0	\$	0
Administration	0	\$\$	\$\$	\$	\$\$	\$\$\$
Storage	0	\$\$	\$\$	\$	0	0
Bandwidth	0	\$\$	\$	\$\$	\$	\$
Resource Management Software	0	0	0	\$	\$	\$
24X7 Support	0	0	0	\$	\$	\$
<b>Total</b>						

Table 1: Cloud TCO Calculation Example (some assumptions are made)

The AWS Economics Center provides all the necessary tools you need to assess your current IT infrastructure. After you have performed a high-level financial assessment, you can estimate your monthly costs using the [AWS Simple Monthly Calculator](#) by entering your realistic usage numbers. Project that costs over a period of 1, 3 and 5 years and you will notice significant savings.

<sup>2</sup> [http://media.amazonwebservices.com/The\\_Economics\\_of\\_the\\_AWS\\_Cloud\\_vs\\_Owned\\_IT\\_Infrastructure.pdf](http://media.amazonwebservices.com/The_Economics_of_the_AWS_Cloud_vs_Owned_IT_Infrastructure.pdf)

## 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.

For more up-to-date information about certifications and best practices, please visit the AWS [Security Center](#).

## Technical and Functional Assessment

A technical assessment is required to understand which applications are more suited to the cloud architecturally and strategically. At some point, enterprises determine which applications to move into the cloud first, which applications to move later and which applications should remain in-house.

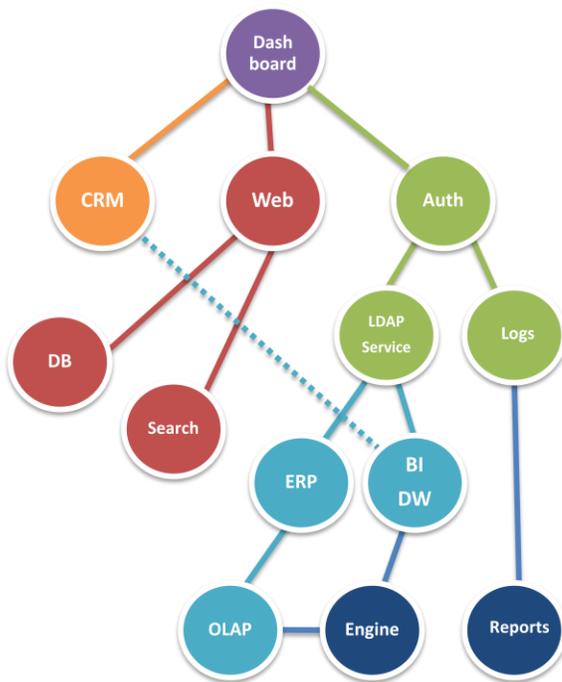
In this stage of the phase, enterprise architects should ask the following questions:

- Which business applications should move to the cloud first?
- Does the cloud provide all of the infrastructure building blocks we require?
- Can we reuse our existing resource management and configuration tools?
- How can we get rid of support contracts for hardware, software and network?

## Create a Dependency Tree and a Classification Chart

Perform a thorough examination of the logical constructs of your enterprise applications and start classifying your applications based on their dependencies, risks, and security and compliance requirements.

Identify the applications and their dependencies on other components and services. Create a dependency tree that highlights all the different parts of your applications and identify their upward and downstream dependencies to other applications. Create a spreadsheet that lists all your applications and dependencies or simply “white-board” your dependency tree that shows the different levels of interconnections of your components. This diagram should be an accurate snapshot of your enterprise application assets. It may look something like the diagram below. It could include all your ERP systems, HR services, Payroll, Batch processing systems, backend billing systems and customer-facing web applications, internal corporate IT applications, CRM systems etc. as well as lower-level shared services such as LDAP servers.



At this stage, you will have clear visibility into your IT assets and you might be able to **classify your applications** into different categories:

- Applications with Top Secret, Secret, or Public data sets
  - Applications with low, medium and high compliance requirements
  - Applications that are internal-only, partner-only or customer-facing
  - Applications with low, medium and high coupling
  - Applications with strict, relaxed licensing
- ...and so on.

Figure 2: Example of whiteboard diagram of all the IT assets and its dependencies (Dependency Tree)

## Identifying the Right “Candidate” for the Cloud

After you have created a dependency tree and have classified your enterprise IT assets, examine the upward and downward dependencies of each application so you can determine which of them to move to the cloud quickly.

For a Web-based application or Software as a Service (SaaS) application, the dependency tree will consist of logical components (features) of the website such as database, search and indexer, login and authentication service, billing or payments, and so on. For backend processing pipeline, there will be different interconnected processes like workflow systems, logging and reporting systems and ERP or CRM systems.

In most cases, the best candidates for the cloud are the services or components that have minimum upward and downward dependencies. To begin, look for systems that have fewer dependencies on other components. Some examples are backup systems, batch processing applications, log processing systems, development, testing and build

systems, web-front (marketing) applications, queuing systems, content management systems, or training and pre-sales demo systems.

To identify which are good candidates for the cloud, search for applications with under-utilized assets; applications that have an immediate business need to scale and are running out of capacity; applications that have architectural flexibility; applications that utilize traditional tape drives to backup data; applications that require global scale (for example, customer-facing marketing and advertising apps); or applications that are used by partners. Deprioritize applications that require specialized hardware to function (for example, mainframe or specialized encryption hardware).

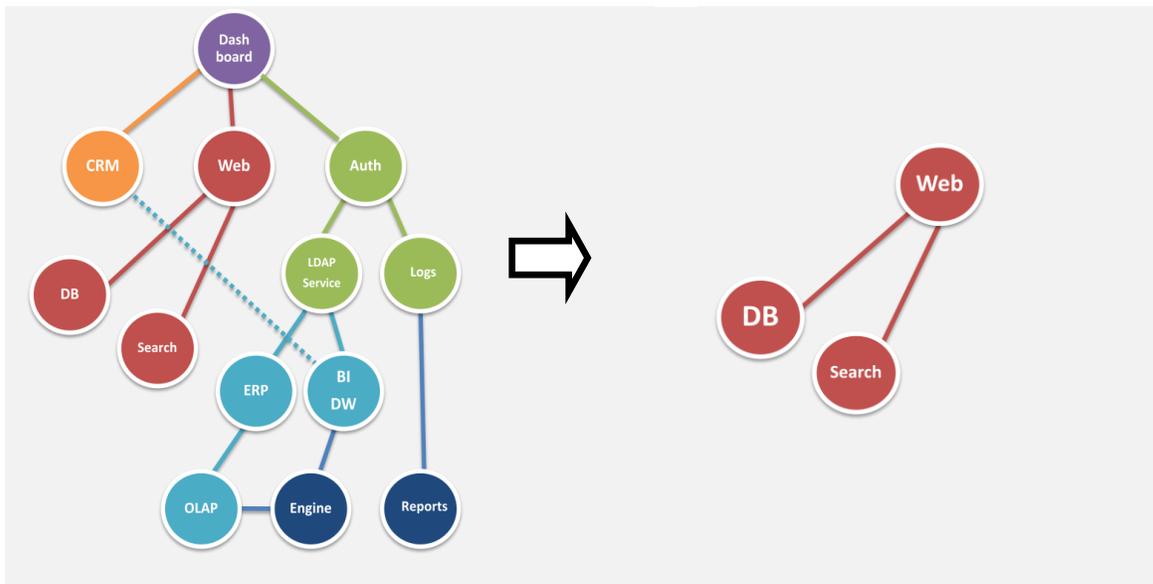


Figure 3: Identify the right candidate for the cloud

Once you have the list of ideal candidates, prioritize your list of applications so that it helps you :

- maximize the exposure in all aspects of the cloud (compute, storage, network, database)
- build support and awareness within your organization and creates highest impact and visibility among the key stakeholders.

Questions to ask at this stage:

- Are you able to map the current architecture of the candidate application to cloud architecture? If not, how much effort would refactoring require?
- Can your application be packaged into a virtual machine (VM) instance and run on cloud infrastructure or does it need specialized hardware and/or special access to hardware that the AWS cloud cannot provide?
- Is your company licensed to move your third-party software used in the candidate application into the cloud?
- How much effort (in terms of building new or modifying existing tools) is required to move the application?
- Which component must be local (on-premise) and which can move to the cloud?
- What are the latency and bandwidth requirements?
- Does the cloud support the identity and authentication mechanism you require?

## Identify the Tools That You Can Reuse

It is important to research and analyze your existing IT assets. Identify the tools that you can reuse in the cloud without any modification and estimate how much effort (in terms of new development and deployment effort) will be required to add “AWS support” to them. You might be able to reuse most of the system tools and/or add AWS support very easily. All AWS services expose standard SOAP and REST Web Service APIs, and provide multiple libraries and SDKs in the programming language of your choice. There are some commercial tools that you won't be able to use in the cloud at this time due to licensing issues, so for those you will need to find or build replacements:

1. **Resource Management Tools:** In the cloud, you deal with abstract resources (AMIs, Amazon EC2 instances, Amazon S3 buckets, Amazon EBS volumes and so on). You are likely to need tools to manage these resources. For basic management, see the [AWS management Console](#).
2. **Resource Configuration Tools:** The AWS cloud is conducive to automation, and as such, we suggest you consider using tools to help automate the configuration process. Take a look at open source tools like Chef, Puppet, and CFEngine, etc.
3. **System Management Tools:** After you deploy your services, you might need to modify your existing system management tools (NOC) so that you can effectively monitor, deploy and “watch” the applications in the cloud. To manage Amazon Virtual Private Cloud resources, you can use the same security policies and use the same system management tools you are using now to manage your own local resources.
4. **Integration Tools:** You will need to identify the framework/library/SDK that works best for you to integrate with AWS services. There are libraries and SDKs available in all platforms and programming languages ([See Resources section](#)). Also, take a look at development productivity tools such as the AWS toolkit for Eclipse.

## Migrating Licensed Products

It is important to iron out licensing concerns during the assessment phase. Amazon is working with many third-party ISVs to smooth the migration path as much as possible. Amazon has teamed with a variety of vendors and is currently offering three different options to choose from:

1. **Bring Your Own License (BYOL)**  
Amazon has teamed with variety of ISVs who have permitted the use of their product on Amazon EC2. This EC2-based license is the most friction-free path to move your software into the cloud. You purchase the license the traditional way or use your existing license and apply it to the product which is available as a pre-configured Amazon Machine Image. *For example, Oracle, Sybase, Adobe, MySQL, JBOSS, IBM and Microsoft have made their software and support available in the AWS cloud using BYOL option.* If you don't find the software that you are looking for in the AWS cloud, talk to your software vendor about making their software available in the cloud. The AWS Business Development Team is available to help you with this discussion.
2. **Use a Utility Pricing Model with a Support Package**  
Amazon has teamed with elite ISVs and they are offering their software as a Paid AMI (using the [Amazon DevPay](#) service). This is a Pay-As-You-Go license in which you do not incur any upfront licensing cost and only pay for the resources you consume. ISVs charge a small premium over and above the standard Amazon EC2 cost which gives you an opportunity to run any number of instances in the cloud for the duration you control. For example, *RedHat, Novell, IBM, Wowza offer pay-as-you-go licenses.* ISVs, typically, also offer a support package that goes with pay-as-you-go license.

### 3. Use an ISV SaaS-based Cloud Service

Some of the ISVs have offered their software as a service and charge a monthly subscription fee. They offer standard APIs and web-based interfaces and are fairly quick to implement. This offering is either fully or partially managed inside the AWS cloud. This option is often the easiest and fastest way to migrate your existing on-premise installation to a hosted on-demand offering by the same vendor or an equivalent offering by a different vendor. In most cases, ISVs or independent third-party enterprise cloud services integrators offer migration tools that can help you move your data. For example, *Mathematica*, *Quantivo*, *Pervasive* and *Cast Iron* provide a SaaS offering based on AWS.

If your enterprise applications are tightly coupled with complex third-party enterprise software systems that have not yet been migrated to the AWS cloud or if you have already invested in multi-year on-premise licensing contracts with the vendor, you should consider refactoring your enterprise applications into functional building blocks. Run what you can in the cloud and connect to the licensed software systems that still run on-premise. Amazon VPC may be used to create an IPSec VPN tunnel that will allow resources running on AWS to communicate securely with resources at the other end of the tunnel in your existing data center. The [whitepaper](#)<sup>3</sup> discusses several ways in which you can extend your existing IT infrastructure to the cloud.

#### Define Your Success Criteria

While you are at this stage, it is important to ask this question: “How will I measure success?”. The following table lists a few examples. Your specific success criteria will be customized to your organization’s goals and culture.

Success Criteria	Old	New	Examples on How to Measure
<b>Cost (CapEx)</b>	\$1M	\$300K	60% savings in CapEx over next 2 years
<b>Cost (OpEx)</b>	\$20K/Year	\$10K/Year	Server-to-Staff ratio improved by 2x 4 maintenance contracts discontinued
<b>Hardware procurement efficiency</b>	10 machines in 7 months	100 machines in 5 minutes	3000% faster to get resources
<b>Time to market</b>	9 months	1 month	80% faster in launching new products
<b>Reliability</b>	Unknown	Redundant	40% reduction in hardware-related support calls
<b>Availability</b>	Unknown	At least 99.99% uptime	20% reduction in operational support calls
<b>Flexibility</b>	Fixed Stack	Any Stack	Not locked into particular hardware vendor or platform or technology
<b>New opportunities</b>	10 projects backlog	0 backlog, 5 new projects identified	25 new projects initiated in 3 months

Table 2: Examples on how to measure success criteria

#### Create a Roadmap and a Plan

By documenting the dependencies, creating a dependency tree, and identifying the tools that you need to build or customize, you will get an idea of how to prioritize applications for migration, estimate the effort required to migrate them, understand the one-time costs involved and assess the timeline. You can construct a cloud migration roadmap. Most companies skip this step and quickly move to the next phase of building a pilot project as it gives a clearer understanding of the technologies and tools.

<sup>3</sup> [http://media.amazonwebservices.com/Extend\\_your\\_IT\\_infrastructure\\_with\\_Amazon\\_VPC.pdf](http://media.amazonwebservices.com/Extend_your_IT_infrastructure_with_Amazon_VPC.pdf)

## 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 a small *greenfield* 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 (See the *Getting Started Center* for more details).

At a minimum, at the end of this stage, you should know how to use the AWS Management Console (or the Firefox plug-ins) and command line tools to do the following:

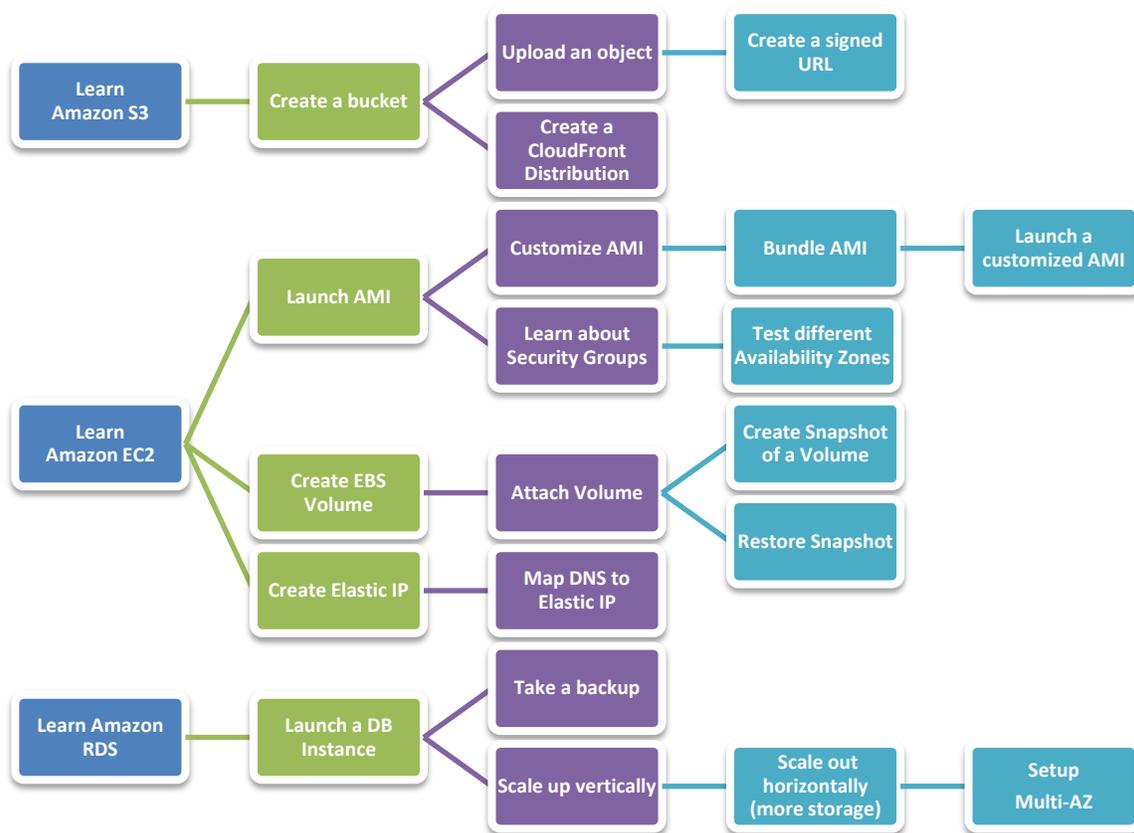


Figure 4: Minimum items to learn about services in a Proof of Concept

### 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 5 highlights the topics you need to learn regarding IAM:

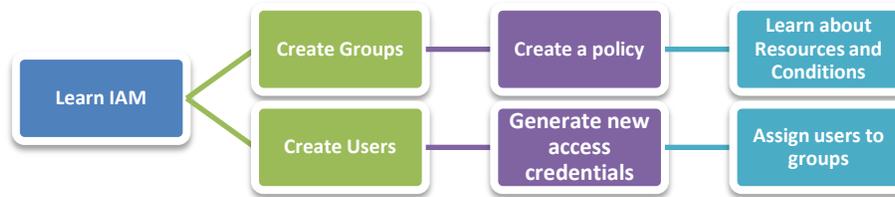


Figure 5: Minimum items to learn about security in a Proof of Concept Phase

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 that 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.

For example, if 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 AMI, 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, AMIs, 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.

## 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?

When choosing the appropriate storage option, one size does not fit all. There are several dimensions that you might have to consider so that your application can scale to your needs appropriately with minimal effort. You have to make the right tradeoffs among various dimensions - cost, durability, query-ability, availability, latency, performance (response time), relational (SQL joins), size of object stored (large, small), accessibility, read heavy vs. write heavy, update frequency, cache-ability, consistency (strict, eventual) and transience (short-lived). Weigh your trade-offs carefully, and decide which ones are right for your application. The beauty about AWS is that it doesn't restrict you to use one service or another. You can use any number of the AWS storage options in any combination.

### 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
<b>Ideal for</b>	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
<b>Ideal examples</b>	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
<b>Not recommended for</b>	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	Clusters
<b>Not recommended examples</b>	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.

## Understand various RDBMS options in the AWS cloud

For your relational database, you have multiple options to choose:

	Amazon RDS	RDBMS AMIs	3 <sup>rd</sup> Party Database Service
<b>RDBMS</b>	MySQL	Oracle 11g, Microsoft SQL Server, MySQL, IBM DB2, Sybase, Informix, PostgreSQL	Vertica, AsterData
<b>Support provided by Managed by AWS</b>	AWS Premium Support Yes	AWS and Vendor No	Vendor No
<b>Pricing Model</b>	Pay-as-you-go	BYOL, Pay-as-you-go	Various
<b>Scalability</b>	Scale compute and storage with a single API call or a click	Manual	Vendor responsibility

Table 4: Relational Database Options

### 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.

## Phase 4: Application Migration Phase

---

In this phase, you should ask the following question:

- How can I move part of or an entire system to the cloud without disrupting or interrupting my current business?

In this phase, you will learn two main application migration strategies: Forklift Migration Strategy and Hybrid Migration Strategy. We will discuss the pros and cons of each strategy to help you decide the best approach that suits your application. Based on the classification of application types (in Phase 1), you can decide which strategy to apply for what type of application.

### 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.

Like with any other migration, having a backup strategy, a rollback strategy and performing end-to-end testing is a must when using this strategy.

### 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.

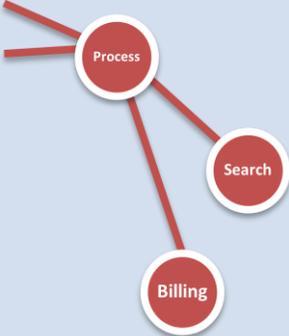
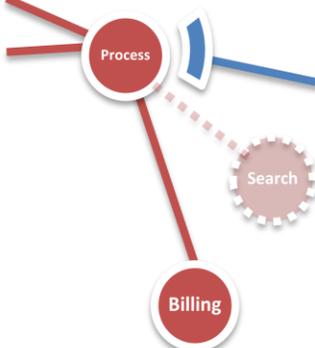
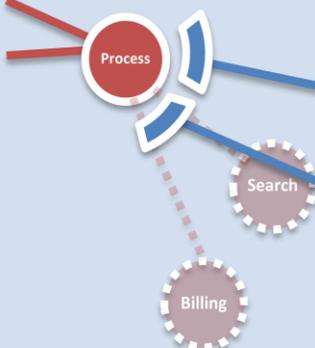
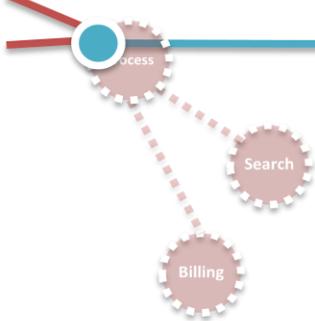
On-site or co-lo	AWS cloud	Notes
		 <p>Service, Business Component, or a Feature that consists of app code, business logic, data access layer and database</p>
		 <p>Thin Layer of “cloud-aware” code to be written that uses web services interface of the component, consists of stubs/skeletons</p>
		<p>Keep the DB close to the component using it: If all the components use the same database, it might be advisable to move all the components and the database together all at once. If all the components use different database instances/schemas and are mutually exclusive but are hosted on the same physical box, it might be advisable to separate the logical databases and move them along with component during migration</p>
		 <p>Proxy may or may not be used.</p>

Table 5: Hybrid Low-risk Migration Strategy of Components into the cloud

In this strategy, you might have to design, architect and build temporary “wrappers” to enable communication between parts residing in your traditional datacenter and those that will reside in the cloud. These wrappers can be made “cloud-aware” and asynchronous (using Amazon SQS queues, wherever applicable) so that they are resilient to changing internet latencies.

This strategy can also be used to integrate cloud applications with other cloud-incompatible legacy applications (Mainframe applications or applications that require specialized hardware to function). In this case, you can write “cloud-aware” web service wrappers around the legacy application and expose them as web service. Since web ports are accessible from outside enterprise networks, the cloud applications can make a direct call to these web services and which in turn interacts with the mainframe applications. You can also setup a VPN tunnel between the legacy applications that reside on-premise and cloud applications.

### **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.

## **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. Questions that you can ask at this stage are:

- Now that I have migrated existing applications, what else can I do in order to leverage the elasticity and scalability benefits that the cloud promises? What do I need to do differently in order to implement elasticity in my applications?
- How can I take advantage of some of the other advanced AWS features and services?
- How can I automate processes so it is easier to maintain and manage my applications in the cloud?
- What do I need to do specifically in my cloud application so that it can restore itself back to original state in an event of failure (hardware or software)?

### **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. The [Migrating to CloudFront<sup>4</sup>](#) whitepaper gives you more information.

### Amazon Elastic MapReduce

For analyzing any large dataset or processing large amount of media, one can take advantage of Amazon Elastic MapReduce. Most enterprises have metrics data to process or logs to analyze or large data sets to index. With Amazon Elastic MapReduce, you can create repeatable job flows that can launch a Hadoop cluster, process the job, expand or shrink a running cluster and terminate the cluster all in few clicks.

### Automate Elasticity

Elasticity is a fundamental property of the cloud. To understand elasticity and learn about how you can build architectures that supports rapid scale up and scale down, refer to the [Architecting for the cloud whitepaper<sup>5</sup>](#). 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.

---

<sup>4</sup> <http://developer.amazonwebservices.com/connect/entry!default.jspa?categoryID=267&externalID=2456>

<sup>5</sup> [http://media.amazonwebservices.com/AWS\\_Cloud\\_Best\\_Practices.pdf](http://media.amazonwebservices.com/AWS_Cloud_Best_Practices.pdf)

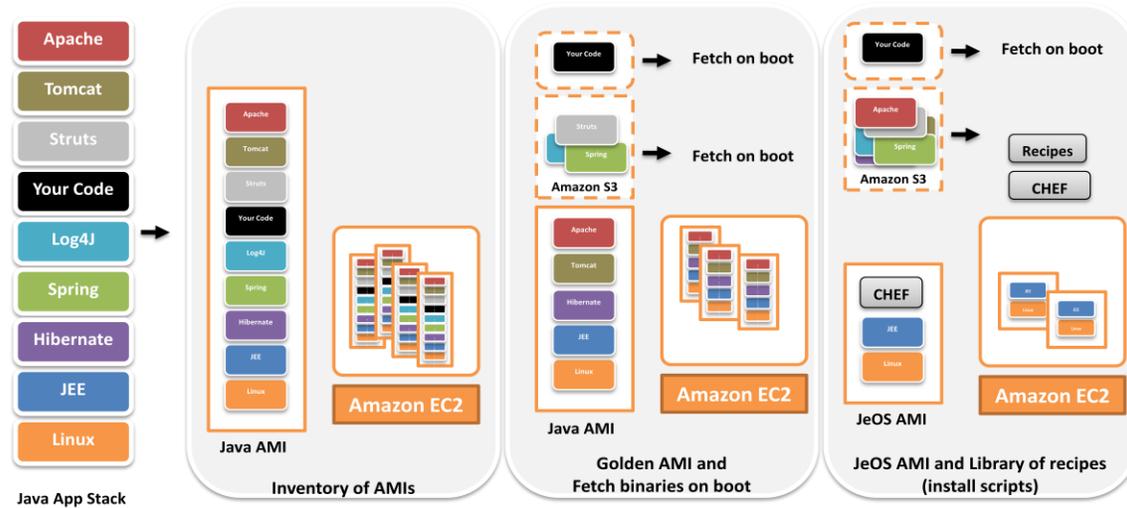


Figure 6: Three ways to automate elasticity while maintaining the upgrade process

## Harden Security

The cloud does not absolve you from your responsibility of securing your applications. At every stage of your migration process, you should implement the right security best practices. Some are listed here:

- Safeguard your AWS credentials
  - Timely rotate your AWS access credentials, and immediately rotate if you suspect a breach
  - Leverage multi-factor authentication
- Restrict users to AWS resources
  - Create different users and groups with different access privileges (policies) using AWS Identity and Access Management (IAM) features to restrict and allow access to specific AWS resources
  - Continuously revisit and monitor IAM user policies
  - Leverage the power of security groups in Amazon EC2
- Protect your data by encrypting it at-rest (AES) and in-transit (SSL)
  - Automate security policies
- Adopt a recovery strategy
  - Create periodic Amazon EBS snapshots and Amazon RDS backups.
  - Occasionally test your backups *before* you need them

## Automate the In-cloud Software Development Lifecycle and Upgrade Process

In the AWS cloud, there is no longer any need to place purchase orders for new hardware ahead of time or to hold unused hardware captive to support your software development lifecycle. Instead, developers, system builders, and testers can request the infrastructure they need minutes before they need it, taking advantage of the vast scale and rapid response time of the cloud. With a scriptable infrastructure, you can completely automate your software development and deployment lifecycle. You could manage your development, build, testing, staging and production

environments by creating re-usable configuration tools, managing specific security groups and launching specific AMIs for each environment.

Automating your upgrade process in the cloud is highly recommended at this stage so that you can quickly advance to newer versions of the applications and also rollback to older versions when necessary. With the cloud, you don't have to install new versions of software on old machines, but instead throw away old instances and re-launch new fresh pre-configured instances. If upgrade fails, you simply throw it away and switch to new hardware with no additional cost.

### **Create a Dashboard of your Elastic Datacenter to Manage AWS Resources**

It should be easy and friction-free for the engineering and project managers to provision and relinquish AWS cloud resources. At the same time, the management team should also have visibility into the ways in which AWS resources are being consumed. The AWS Management Console provides a view of your cloud datacenter. It also provides you with basic management and monitoring capabilities (by way of Amazon CloudWatch) for your cloud resources. The AWS Management Console is continually evolving. It offers rich user interface to manage AWS services. However, if the current view does not fit your needs, we advise you to consider using third party tools that you are already familiar with (like CA, IBM Tivoli) or to create your own console by leveraging the Web Service APIs. Using Web Service APIs, it's fairly straightforward to create a web client that consumes the web services API and create custom control panels to suit your needs. For example, if you have created a pre-sales demo application environment in the cloud for your sales staff so that they can quickly launch a preconfigured application in the cloud, you may want to create a dashboard that displays and monitors the activity of each sales person and each customer. Manage and limit access permissions based on the role of the sales person and revoke access if the employee leaves the company.

There are several libraries available in our [Resource Center](#) that can help you get started with creating the dashboard that suits your specific requirement.

### **Create a Business Continuity Plan and Achieve High Availability (Leverage Multiple Availability Zones)**

Many companies fall short in disaster recovery planning because the process is not fully automatic and because it is cost prohibitive to maintain a separate datacenter for disaster recovery. The use of virtualization (ability to bundle AMI) and data snapshots makes the disaster recovery implementation in the cloud much less expensive and simpler than traditional disaster recovery solutions. You can completely automate the entire process of launching cloud resources which can bring up an entire cloud environment within minutes. When it comes to failing over to the cloud, recovering from system failure due to employee error is the same as recovering from an earthquake. Hence it is highly recommended that you have your business continuity plan and set your Recovery Time Objective (RTO) and Recovery Point Objective (RPO).

Your business continuity plan should include:

- data replication strategy (source, destination, frequency) of databases (Amazon EBS)
- data backup and retention strategy (Amazon S3 and Amazon RDS)
- creating AMIs with the latest patches and code updates (Amazon EC2)
- recovery plan to fail back to the corporate data center from the cloud post-disaster

The beauty of having a business continuity strategy implemented in the cloud is that it automatically gives you higher availability across different geographic regions and Availability Zones without any major modifications in deployment and data replication strategies. You can create a much higher availability environment by cloning the entire architecture and replicating it in a different Availability Zone or by simply using Multi-AZ deployments (in case of Amazon RDS).

## Phase 6: Optimization Phase

---

In this phase, you should focus on how you can optimize your cloud-based application in order to increase cost savings. Since you only pay for the resources you consume, you should strive to optimize your system whenever possible. In most cases, you will see immediate value in the optimizations. A small optimization might result in thousands of dollars of savings in your next monthly bill. At this stage, you can ask the following questions:

- How can I use some of the other AWS features and services in order to further reduce my cost?
- How can I improve the efficiency (and reduce waste) in my deployment footprint?
- How can I instrument my applications to have more visibility of my deployed applications? How can I set metrics for measuring critical application performance?
- Do I have the necessary cloud-aware system administration tools required to manage and maintain my applications?
- How can I optimize my application and database to run in more elastic fashion?

### 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.

### Improve Efficiency

The AWS cloud provides utility-style pricing. You are billed only for the infrastructure that has been used. You are not liable for the entire infrastructure that may be in place. This adds a new dimension to cost savings. You can make very measureable optimizations to your system and see the savings reflected in your next monthly bill. For example, if a caching layer can reduce your data requests by 80%, you realize the reward right in the next bill.

Improving performance of the application running in the cloud might also result in overall cost savings. For example, if your application is transferring a lot of data between Amazon EC2 and your private data center, it might make sense to

compress the data before transmitting it over the wire. This could result in significant cost savings in both data transfer and storage. The same concept applies to storing raw data in Amazon S3.

## Management and Maintenance

### Advanced Monitoring and Telemetry

Implement telemetry in your cloud applications so it gives you the necessary visibility you need for your mission-critical applications or services. It is important to understand that end-user response time of your applications depends upon various factors, not just the cloud infrastructure – ISP connectivity, third-party services, browsers and hops, just to name a few. Measuring and monitoring the performance of your cloud applications will give you the opportunity to proactively identify any performance issues and help you diagnose the root causes so you take appropriate actions. For example, if an end-user accessing the nearest node of your globally hosted application is experiencing a lower response rate, perhaps you can try launching more web servers. You can send yourself notifications using Amazon Simple Notifications Service (HTTP/Email/SQS) if the metric (of a given AWS resource or an application) approaches an undesired threshold.

### 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.

### Re-engineer your application

To build a highly scalable application, some components may need to be re-engineered to run optimally in a cloud environment. Some existing enterprise applications might mandate refactoring so that they can run in an elastic fashion. Some questions that you can ask:

- Can you package and deploy your application into an AMI so it can run on an Amazon EC2 instance? Can you run multiple instances of the application on one instance, if needed? Or can you run multiple instances on multiple Amazon EC2 instances?
- Is it possible to design the system such that in the event of a failure, it is resilient enough to automatically re-launch and restart?
- Can you divide the application into components and run them on separate Amazon EC2 instances? For example, can you separate a complex web application into individual components or layers of Web, App and DB and run them on separate instances?
- Can you extract stateful components and make them stateless?
- Can you consider application partitioning (splitting the load across many smaller machines instead of fewer larger machines)?
- Is it possible to isolate the components using Amazon SQS?
- Can you decouple code with deployment and configuration?

## Decompose your Relational database

Most traditional enterprise applications typically use a relational database system. Database administrators often start with a DB schema based on the instructions from developer. Enterprise developers assume unlimited scalability on fixed infrastructures and develop the application against the schema. Developers and database architects may fail to communicate with each other on what type of data is being served, which makes it extremely difficult to scale that relational database. As a result, much time may be wasted migrating data to a “bigger box” with more storage capacity, or scaling up to get more computing horsepower. Moving to the cloud gives them the opportunity to analyze their current relational database management system and make it more scalable as a part of the migration. Some techniques that might help take the load off of your RDBMS:

- Move large blob object and media files to Amazon S3 and store a pointer (S3 key) in your existing database
- Move associated meta-data or catalogs to Amazon SimpleDB
- Keep only the data that is absolutely needed (joins) in the relational database
- Move all relational data into Amazon RDS so you have the flexibility of being able to scale your database compute and storage resources with an API call *only when you need it*
- Offload all the read load to multiple Read Replicas (Slaves)
- Shard (or partition) the data based on item IDs or names

## Implement Best Practices

Implement various best practices highlighted in the [Architecting for the cloud whitepaper](#). These best practices will help you to create not only a highly scalable application conducive to the cloud but will also help you to create a more secure and elastic application.

## Conclusion

The AWS cloud brings scalability, elasticity, agility and reliability to the enterprise. To take advantage of the benefits of the AWS cloud, enterprises should adopt a phase-driven migration strategy and try to take advantage of the cloud as early as possible. Whether it is a typical 3-tier web application, nightly batch process, or complex backend processing workflow, most applications can be moved to the cloud. The blueprint in this paper offers a proven step by step approach to cloud migration. When customers follow this blueprint and focus on creating a proof of concept, they immediately see value in their proof of concept projects and see tremendous potential in the AWS cloud. After you move your first application to the cloud, you will get new ideas and see the value in moving more applications into the cloud.

## Further Reading

1. Migration Scenario #1: [Migrating web applications to the AWS cloud](#)
2. Migration Scenario #2: [Migrating batch processing applications to the AWS cloud](#)
3. Migration Scenario #3: [Migrating backend processing pipelines to the AWS cloud](#)