Understanding Your Application Readiness when Migrating to AWS

October 2020

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents current AWS product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided “as is” without warranties, representations, or conditions of any kind, whether express or implied. The responsibilities and liabilities of AWS to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Contents

Abstract

When a customer is ready to migrate their applications from on-premises to AWS, they often have questions:

- What applications can be migrated to AWS?
- How complex is the migration?
- How do I prioritize the migration workload?

This whitepaper presents an effective approach for investigating the *fitness* of the applications to run on AWS, understanding the *complexity* of the application migration, and organizing the *prioritization* of applications into different waves of migration to AWS.

Introduction

A recent whitepaper by IDC shows that customers who migrate to AWS can experience 51% reduced costs of operations, 62% increased IT staff productivity, and 94% reductions in downtime. Migrations to AWS include moving any workload from an on-premises environment, hosting facility, or other public cloud. AWS works with thousands of organizations to migrate workloads such as applications, websites, databases, storage, physical or virtual servers, and entire data centers.

Workload migration to AWS has become the new normal. Companies of every size have realized the benefits of the cloud. For most organizations, the question isn't "if" anymore, it's when and how, and more specifically:

- Which applications can we move to AWS?
- What is the complexity and effort involved in moving those applications?
- Which applications should we migrate first?

However, there is no single tool available to help you identify the fitness of an application to be migrated from on-premises to the AWS Cloud. We must engage different AWS partners to conduct the application assessment.

This whitepaper helps you understand the components of your application's readiness to migrate your application to AWS. It discusses a methodology that includes three different perspectives on fitness, complexity, and prioritization, to give you a holistic view of how to accelerate your application migration to AWS.

Methodology

First, you need to understand the fitness of the application running on the AWS Cloud. it evaluates the application constraints when it runs on AWS Cloud with respect to six different dependencies:

- Security and data sovereignty
- Hardware platform
- Operating System (OS) compatibility
- Storage
- Network
- Application

Next, the outcomes of the application fitness analysis, which is covered in detail in the next section of this whitepaper, are used as the considerations to determine the complexity of the application to be migrated to the AWS Cloud. We will use the 6Rs approach to understand the complexity of migration patterns. In this methodology, there are six different migration patterns:

- **R**etain – There are some applications that are not suitable for migration to the AWS Cloud. This can be due to data sovereignty or unresolved non-x86 dependencies. These applications are left in place.
- **R**etire – Applications are no longer useful to the business and can be retired.
- **R**ehost – Applications are migrated to the AWS Cloud without making any change to the components (application code, database, or other components) of the application.
- **R**eplatform – The applications being migrated to AWS involve software version upgrades or configuration changes to adapt to the new architecture platform on AWS cloud.
- **R**epurchase – If the current version of the application is not optimized for deployment on the AWS Cloud, or the current terms and conditions don't allow a bring-your-own-license (BYOL) model to the AWS Cloud, you can move to a Software as a Service (SaaS) model or to off-the-shelf software by Independent Software Vendors (ISV).

- **R**efactor/**R**e-architect – The application is re-architected and rebuilt with a new code base to utilize cloud native features. This migration pattern is used primarily when the business objective is to improve agility, scalability, and performance of the application that is otherwise not achievable with the current architecture.

Finally, you evaluate the prioritization or wave to migrate the application from your on-premises data center to AWS. You can determine prioritization of the application moving to AWS based on the following factors:

- Acceleration
- Complexity
- Business criticality
- Revenue impact
- Hardware refreshment
- Access pattern

Holistic View of Analysis

In this section we will analyze those dimensions and attributes that are used to understand the application readiness for moving to AWS with respect to fitness, complexity and prioritization.

Fitness of Application

Today, most modern applications are developed to the The Twelve-Factor App standard, which makes them good candidates to run on AWS using distributed computing. However, there are some dependencies from legacy applications which should be mitigated before migrating to AWS.

Dependencies

These dependencies are common considerations used to determine if an application can be migrated to AWS. Each dependency listed has a table containing a checklist to identify the constraints of the dependency, scenarios under which you would take the checklist into consideration, and recommendations.

Security and Data Sovereignty

Table 1 – Security and data sovereignty checklists, scenarios, and recommendations

Security Checklists	Scenarios	Recommendations
Are there any specific requirements for the data to remain at an on-premises datacenter or in-country due to mandatory compliance policies from local regulators or internal governance control?	Data sovereignty comes into play when an organization's data is stored outside of their datacenter or country, and is subject to the laws of the country in which the data resides. The main concern with data sovereignty is maintaining privacy regulations and keeping foreign countries from being able to subpoena data.	• If the application consists of sensitive data that must be stored in an on-premises datacenter or in-country due to local regulators or internal compliance policies, the application must run at an on-premises datacenter. • If the application and database are running on separated instances, consider whether it's possible to break the tier of application by keeping the database layer on-premises but migrating the application logic layer to AWS.

Hardware Platform

Table 2 – Hardware platform checklists, scenarios, and recommendations

Hardware Checklists	Scenarios/Examples	Recommendations
Does your application (App) or database (DB) use hardware based on the x86 architecture?	AWS does not support non-x86 platforms such as IBM Z Mainframe, Unisys ClearPath, VAX stations, HP 3000 and HP 9000, IBM Power Systems, Sun SPARC, and others. See Supported operating systems.	• To migrate applications running on non-x86 to AWS, check with the software vendor or internal application developers to learn if the application can run on a heterogeneous platform instead. • If the application is not a good candidate for the heterogeneous platform, ask the software vendor or developers if they can

		rewrite or re-compile the code to a suite that runs on the x86 platform instead. • Alternatively, explore using hardware emulation for legacy systems that can be re-hosted on AWS using cross-platform hypervisors from Stromasys, an AWS Partner Network (APN) Standard Technology Partner. See this blog post for more information.
Is your application running on a hardware appliance-based solution, or tightly coupled to the underlying hardware?	Hardware-appliance refers to a device that is dedicated to a specific function in contrast to a general-purpose computer. The hardware comes pre-installed with its operating system and application. For example, Oracle Exadata is a typical hardware appliance.	• AWS does not support any 3rd party hardware appliances for migration to the AWS platform. • If the application is running on a 3rd party hardware appliance, ask the software vendor if they offer any software appliance that runs on Amazon Elastic Compute Cloud (Amazon EC2). For example, both Pivotal Greenplum and HP Vertica offer a software appliance version that runs on the AWS Cloud. • To migrate tightly coupled hardware appliances such as Oracle Exadata to AWS, consider following steps: 1. Performance baselining for on-premises and AWS 2. Infrastructure right sizing 3. Database fine tuning and optimization 4. Application tuning • See this blog post to read the journey of Australia

		Finance Group's migration of their Oracle Exadata to AWS.
Is your application required to use a USB key or dongle license that is attached to the server hardware for licensing validation purposes?	A USB or dongle license offers dongle-based copy protection that ties the application to a physical device attached to the server's USB or COM port.	• AWS does not support Bring Your Own Device (BYOD) on the AWS platform. • If your application requires a USB key or dongle attached to server hardware for licensing validation purposes, ask your software vendor if they offer a software version that can support the AWS platform.
Does your application require connection to an Ethernet Media Access Control (MAC) address to generate a valid license file or key?	MAC address licensing is a popular and simple way to license software products. The software needs your server's MAC address to generate the license file or key.	As of the date of this writing, Amazon EC2 does not support manual assignment of a MAC address. You must re-provision (re-install and re-configure) the application to generate a new license key when you migrate to Amazon EC2 running on an AWS environment under the guidance and support of the software vendor.

Operating System (OS) Compatibility

Table 3 – Operating System checklists, scenarios, and recommendations

Operating System Checklists	Scenarios	Recommendations
Does your App/DB support environments (Windows or Linux) with the operating systems that are supported on AWS?	• Amazon EC2 supports the migration of different versions of Microsoft Windows Server and Linux (Red Hat, CentOS, Ubuntu, SUSE, Kali, Oracle Linux, Debian, Fedora, and Amazon Linux) to AWS. However, you must ensure	• If your App/DB is running on an OS that is not supported by the AWS platform, check with the software vendor or application developers to see if they can support an OS that is compatible with the AWS platform.

	that the current OS version for your application is supported by AWS. See this documentation page to view OS versions which can be imported to and exported from Amazon EC2. • If you are running a supported OS version on AWS that is declared End of Support (EOS) by the software vendor, you will not receive any security and software updates. For example, Microsoft ended extended support for Windows Server 2003 on July 14, 2015. If you are running Windows Server 2003 on AWS, this will put your applications and business at risk.	• If you plan to migrate your App/DB to Amazon EC2 without changing the OS/DB/App version, you can leverage AWS CloudEndure Migration to automate and accelerate the migration process. • Alternatively, if you plan to upgrade an OS version that is compatible with the AWS platform, ask your App/DB software vendors if the newer OS version can support the current App/DB version without causing compatibility issues. • If your application running on Windows 2003/2008 is not compatible with newer Windows OS versions on the AWS platform, consider the AWS End-of-Support Migration Program (EMP) for Windows Server without any refactoring. The EMP technology identifies the dependencies that your app has on the outdated OS, and creates a package that includes the resources necessary for the apps to run on the newer version of Windows Server.

Storage

Table 4 – Storage checklists, scenarios, and recommendations

Storage Checklists	Scenarios	Recommendations
Does your application depend on shared	For example, your application and	• As of the date of this writing, most of the AWS regions do not support IP

storage for High Availability (HA) clustering with the IP multicast protocol?	database are running on Red Hat Cluster Suite (RHCS) with shared storage using IP multicast for heartbeat health check purposes.	multicast workloads and Amazon Multi-Attach EBS. • If you intend to migrate your application with HA provisioning using shared storage with IP multicast to AWS, consider a 3rd party mechanism with application-aware availability such as SIOS or Veritas InfoScale from AWS Marketplace. • For databases using HA clustering with shared storage and IP multicast on-premises, we recommend that you migrate to AWS using Amazon Relational Database Service (RDS) with Multi-AZ deployment. This is a managed database service that provides automated failover with synchronous replication. Consider using AWS Database Migration Service (DMS) to automate the migration process. • If your application does not support Amazon RDS, consider native replication features from the application's database engine to deliver similar HA capability on AWS. For example, Master HA (MHA) for MySQL, Always On for Microsoft SQL Server, or HADR for IBM DB2.
Is there any data replication between production and Disaster Recovery (DR) sites using a shared storage feature?	For example, you use storage arrays such as EMC, HP, Dell, IBM, or Hitachi to replicate your application data from the production datacenter to a remote datacenter for DR purposes.	• As the date of this writing, Amazon Elastic Block Store (Amazon EBS) does not support data replication across two different AWS Availability Zones (AZs) for DR purposes. • If you are planning to have DR across two different AWS AZs or AWS regions after migrating to AWS, consider using AWS CloudEndure DR.

Network

Table 5 – Network checklists, scenarios, and recommendations

Network Checklists	Scenarios	Recommendations
Does your application have any specific low network latency access from other applications placed in on-premises?	• Latency refers to the amount of time it takes to send a packet of data from one application to another application. • In financial and telecommunication industries, there are often real-time transactions and billing applications that require low network latency to ensure a fast performance response.	If the application to be migrated to AWS is sensitive to network latency regarding upstream and downstream applications within the same datacenter, consider migrating the application and all the upstream and downstream applications to avoid slow performance response issues.
Does your application need to support IP multicast protocol?	• For example, middleware such as Oracle WebLogic or Red Hat JBoss multicast deployment uses standard User Datagram Protocol (UDP) multicast to broadcast cluster messages to a group that is listening on the multicast address and port over which the message is sent. • The market data exchange application from the Stock Exchange commonly uses multicast protocol to avoid bandwidth congestion.	• As of the date of this writing, most of the AWS regions do not support IP multicast workloads. • If your application is running based on IP multicast, ask the software vendor if they can support IP unicast instead. For example, Oracle WebLogic and Red Hat JBoss can support a cluster running either multicast or unicast. • If your application does not support unicast with an option to rewrite or recompile the code, consider running

		your application by setting up a Generic Routing Encapsulation (GRE) tunnel on AWS. This is a tunneling protocol developed by Cisco Systems that can encapsulate a wide variety of network layer protocols inside virtual point-to-point links or point-to-multipoint links over an Internet Protocol network.
If your application is running behind the load balancer, does it support passive mode File Transfer Protocol (FTP) traffic?	It is possible that an application requires multi-protocol support through a single load balancer. For example, if your application requires support from FTP and HTTP from the same hostname, then your load balancer will need to support both protocol and port combinations.	• The load balancers from the Amazon Elastic Load Balancer (ELB) family (Classic Load Balancer (CLB), Network Load Balancer (NLB), and Application Load Balancer (ALB)) do not support FTP protocol. • If your application requires a fronted load balancer to support FTP, consider a 3rd party solution from AWS Marketplace such as F5.
Is your application running behind a load balancer using an algorithm of Weighted	Check with the application owner or infrastructure person in charge to understand the algorithm being	• All the load balancers from the Amazon ELB family support only Round-

Round-Robin or Least Connection to distribute incoming requests to backend targets?	configured for your existing load balancer.	Robin algorithms with the exception of ALB, which supports Least Outstanding Request (LOR) with HTTP/HTTPS protocol. See Elastic Load Balancing Features for details. • If your application requires a fronted load balancer to support algorithms such as Weighted Round-Robin or Least Connect with protocols other than HTTP/HTTPS, consider a 3rd party solution from the AWS Marketplace such as NGINX.
Is your application using a load balancer to perform request inspection for payload scanning?	If an application is performing payload scanning, the load balancer scans the entire request (headers and body) for patterns. When patterns are identified, the load balancer performs an action on the request based on the type of pattern found. For example, a load balancer could use payload scanning to identify and block any requests that contain a forbidden keyword in the body of the request. Payload scanning is a compute-intensive process, and can be used for many purposes such as protection against common attack patterns (SQL injection, XSS scripting, and others), allow-listing and deny-listing requests, enforcing data quality standards, and more.	• As of the date of this writing, Amazon ELB does not support payload scanning. • If your application requires a fronted load balancer to support payload scanning, consider a 3rd party solution from AWS Marketplace such as NGINX.

Does your application require a load balancer for direct manipulation of the HTTP header?	Some applications require load balancers to perform direct manipulation of HTTP requests for incoming traffic before proxying these requests to application nodes. These manipulations can add, remove, or rewrite parts of the request.	• As of the date of this writing, Amazon ELB does not support direct HTTP header manipulation. • If your application requires a fronted load balancer to support direct HTTP request manipulation, consider a 3rd party solution from AWS Marketplace such as NGINX.
Does your application generate high volume (more than one terabyte) of data to be transferred to other applications for processing every day?	You may have an application that generates a high volume of data to be transferred to a data analytic application, data lakes, or other applications for further analysis or batch processing purposes.	• All data transferred out from Amazon Virtual Private Cloud (Amazon VPC) is charged accordingly. See Your Comprehensive Guide to Understanding AWS Data Transfer Costs. • If your application generates a high volume of egress data from a VPC to an on-premises datacenter for data analytic purpose, consider migrating your data analytic tool to AWS. This can reduce bandwidth utilization and data transfer cost, and deliver better workload performance.

Application

Table 6 – Application checklists, scenarios, and recommendations

Application Checklists	Scenarios	Recommendations
Is the application currently in use by the business?	Ask the application owner if the application will be decommissioned soon, or if it will be replaced by newly developed or purchased software.	• If your application is not actively used by the business and is under decommission program scope, consider retiring this application without spending the additional effort to migrate it to AWS. • If your application must be kept for data archiving after retirement to meet regulatory compliance, you can migrate the application to AWS using minimal resources.
Does your application have any IP addresses that are hard coded in the source code?	The software vendor or application developer may hardcode the IP address of the remote end point into the source code without having hostname or DNS resolution.	• *Avoid using hardcoded addresses in code.* • When you migrate your application to AWS, you are required to change the IP address in accordance with new classless inter-domain routing (CIDR) assigned to VPCs on AWS. This is necessary to avoid IP overlapping that will cause IP routing issues between the on-premises datacenter and AWS. • If your application has any IP hard coding in the source, invite your application developer to change the IP address in the source code by using DNS resolution instead. • If you have specific reasons that require you to migrate

		an on-premises virtual machine (VM) to AWS while preserving its IP address, consider using the AWS Migration Hub (part of the AWS Server Migration Service) and Aviatrix IPmotion, which enables IP address preservation after a VM is migrated to AWS via the AWS Server Migration Service.
Is your application certified compatible to run on AWS EC2 instances?	For example, SAP has certified only certain instance types from the Amazon EC2 family which are recommended for production environment purposes.	Ask your software vendor if your application must run on specific Amazon EC2 instance types for a production environment. This is essential to ensure that the applications will always deliver the best and most consistent performance to meet business requirements.
Is your application using any CI/CD pipeline for automated provisioning and code release management?	N/A	If your application relies on a CI/CD pipeline to automate application provisioning and release, ensure that the CI/CD infrastructure is migrated to AWS before you migrate your application.
Will your application vendor and/or operation support team continue to support the application if it is moved to the AWS Cloud?	Different software vendors or/and internal operation teams might have different concerns about technology architecture, technical skillsets, and operation processes to support their application running on AWS.	Ask your software vendor and application operational team if they will continue to provide operational support after you migrate your application to AWS.
Is the license transferable from on-premises to AWS?	Every software vendor has their own license policy to bring their license to AWS. For example, Microsoft only allows their	Ask your software vendor if they allow you to bring your current license from on-premises to AWS.

	users to bring their OS license to Amazon Dedicated Host.

Complexity of Migration Patterns

To understand your complexity of migration, you must receive the outputs from Fitness of Application as inputs to determine the right migration pattern for your applications. Table 7 shows which migration pattern can be used to migrate your applications to AWS, with examples of scenarios.

Table 7 – 6R migration pattern analysis

Migration patterns	Pattern description	Examples of scenarios
Retain	• You will keep the application in your on-premises environment. • Minimal analysis and validation of scope and application affinity. • Dependency on integrating service management.	• Unresolved security (data sovereignty) and physical dependencies. • Non-x86 UNIX (Mainframe/IBM AIX/Sun Solaris) applications without option for Replatform and Refactor/Re-architect. • License restriction to move the application to AWS due to dongle or USB key attachment to Amazon EC2.
Retire	• Applications will be decommissioned within three to six months. • No migration to AWS environment. • Require application owners to acknowledge and sign off.	• Within existing decommission program scope.

Rehost	• Redeploy an application's components to another physical, virtual, or cloud location without:
	  ○ Recompiling
	  ○ Altering the OS/DB/Application version
	  ○ Altering the application code
	  ○ Modifying features and functions
	• Very minimal or no change of configuration to make the application work on AWS.
		• Virtual to Virtual (V2V) or Physical to Virtual (P2V).
		• Red Hat Enterprise Linux (RHEL) 6 or above (64-bit).
		• Windows 2008 and above.

Migration patterns	Pattern description	Examples of scenarios
	• You can migrate workloads to AWS by using automation tools such as CloudEndure Migration. • User Acceptance Test (UAT) and System Integration Testing (SIT) – some level of application testing.	
Replatform	• Change or upgrade of the OS/DB/application version onto the AWS Cloud. • Make minimal changes to code and configurations to adapt to the new architecture platform, but don't change the code structure or the features and functions it provides. • You are required to reprovision the OS/DB/Application before or after migrating to AWS. • UAT and SIT is highly recommended.	• W2K/W2K3/W2K8 to Windows 2012 or above. • RHEL 32-bit or RHEL 5.0 and below. • Migrate from IBM AIX/Sun Solaris to Linux. • All clusters using shared storage with multicast protocol. • Change the IP address in the source code by maintaining OS, DB, and application versions. • Change application configurations from multicast to unicast by maintaining OS, DB, and application versions. • Migrate your database to Amazon RDS using a homogenous database engine. • Re-provision your container on AWS by using Amazon Elastic Kubernetes Service (Amazon EKS) or Amazon Elastic Container Service (Amazon ECS).

Migration patterns	Pattern description	Examples of scenarios
Repurchase	• Repurchase the capability via a different channel. • Require deep engagement with business stakeholders to understand and embrace new business requirements. • Potentially involve data conversion when migrating to AWS. • UAT and SIT is mandatory.	• On-premises customer relationship management (CRM) to Salesforce.com.
Refactor/ Re-architect	• **Refactor –** restructure and optimize existing code without changing its external behavior to remove technical debt and to improve the component's features and structure. • **Re-architect –** Materially alter the application code so you can shift it to a new application architecture and fully exploit AWS cloud native services. • UAT and SIT is mandatory.	• Application porting from mainframe to x86 platform. • Modernize the application architecture from monolithic to microservices by using AWS cloud native services. • Commercial to open source database migration – migrate Oracle/Microsoft SQL database to an open source database such as Amazon Aurora, MySQL, PostgreSQL, or MariaDB. Consider using AWS Database Migration Service (AWS DMS) to automate the heterogeneous database migration with minimum downtime. • Repurpose the database for optimizing data structure purposes – relational, NoSQL (key value pair), document. Graph, data warehouse, ledger, and cache.

Table 8 indicates the relative comparison of different migration patterns with respect to complexity, time/velocity, cost and optimization. The complexity or change effort of each migration pattern varies from the easiest (Retire and Retain) to the most difficult (Refactor/Re-architect). In most enterprise migration, 70% to 80% of the applications are candidates for Rehost and Replatform. This gives you high velocity to move your applications to AWS. The huge effort and high complexity of Refactor/Re-architect will cost you more as compared to other migration patterns. However, Refactor/Re-architect will give you better value return on application optimization when you migrate to AWS, because you can include new business features to drive better operational efficiency, resiliency, business agility, user productivity, and customer obsession.

Table 8 – Relative comparison of 6R migration patterns

Migration patterns	Complexity	Time/Velocity	Cost	Optimization
Retain	-	-	-	-
Retire	♦	♠	$	-
Rehost	♦♦	♠♠	$$	-
Replatform	♦♦♦	♠♠♠	$$$	♣
Repurchase	♦♦♦♦	♠♠♠♠	$$$$	♣♣♣
Refactor/Re-architect	♦♦♦♦♦	♠♠♠♠♠	$$$$$	♣♣♣♣♣

Prioritization of Migration Waving

The order of sequence to move your application to AWS varies in accordance with your business requirements. For example:

- If you are conservative and want to start with easier workloads to drive a quick win on AWS migration without a huge revenue impact and business risk, start with Rehost and Replatform migration patterns.
- If you plan to migrate an application which is hosted on hardware and which will no longer be supported soon, you might want to prioritize migration of this application to AWS first.
- If you are new to AWS and would like to learn and acquire more AWS knowledge, consider starting your migration journey with applications that have low business impact and criticality to minimize business risks.

There is no right or wrong answer to establish your migration waves. Your strategy depends on your business priorities with combinations of different factors, as shown in Table 9.

Table 9 – Factors affecting migration prioritization

Factors	Checklist	Recommendations
Acceleration	How can I accelerate my applications migration to AWS?	• Start the migration with low complexity applications. • Start the migration with Rehost followed Replatform, and then Refactor.
Complexity	What are the upstream and downstream impacts if I migrate this application to AWS?	Consider migrating this application with other downstream and upstream applications to resolve slow network latency response issues.
Business criticality	Is this a business critical application that will cause a serious financial, legal, or productivity loss during migration to AWS?	To minimize the business impact when you first migrate your application to AWS, consider placing critical business applications in later migration phases.

Factors	Checklist	Recommendations
Revenue impact	Is this a customer facing application which will cause revenue and customer experience impact?	To minimize the revenue and customer experience impact when you first migrate your application to AWS, consider placing applications with high revenue impact in later migration phases.
Hardware refreshment	Is this application hosted on hardware which will lose support soon?	Migrate this application first regardless of its migration pattern consideration.
Access pattern	What is the access pattern of this application, 7x24 (Monday to Sunday by 24-hour) or 5x8 (Monday to Friday by 8 hours)?	If this application must be accessed by users in 7x24 with no long hours (> 4-8 hours) of down time allowed, consider waiting for a long weekend public holiday window to schedule this migration.

Conclusion

The systematic methodology discussed in this whitepaper gave you a better understanding of your application readiness when you to migrate from an on-premises datacenter to the AWS Cloud. It walked you through the dependencies, patterns, and factors to consider during migration regarding the aspects of fitness, complexity and prioritization. This methodology has been used to accelerate the planning and execution phases for many real-world large scale cloud migration projects.

Before you migrate your application to AWS, ensure that a well-defined AWS Landing Zone is in place. The landing zone should include:

- AWS account structure
- Networking – direct connect, VPN, DNS, NTP, DHCP, subnet and VPC design and routing
- Security – identity access management, detective control, infrastructure security, and data protection
- Logging
- Monitoring
- Backup and restore

For more information, see Mobilize your organization to accelerate large-scale migrations.

It is also essential to ensure that cloud operation support and processes (incident, problem, configuration, and change management) are defined and approved by a Cloud Center of Excellence (CCoE). For more information, see Using a Cloud Center of Excellence (CCOE) to Transform the Entire Enterprise.

Contributors

Contributors to this document include:

- CK Tan, Principal Specialist Solution Architect (SA) – Migration for APAC region

Document Revisions

Date	Description
October 2020	First publication