AWS Cloud Adoption Framework

Platform Perspective

November 2015

Notices

This document is provided for informational purposes only. It represents AWS's current product offerings and practices as of the date of issue of this document, which are subject to change without notice. Customers are responsible for making their own independent assessment of the information in this document and any use of AWS's products or services, each of which is provided "as is" without warranty of any kind, whether express or implied. This document does not create any warranties, representations, contractual commitments, conditions or assurances from AWS, its affiliates, suppliers or licensors. 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

The Amazon Web Services (AWS) Cloud Adoption Framework (CAF)[1] provides best practices and prescriptive guidance to accelerate an organization's move to cloud computing. The CAF guidance is broken into a number of areas of focus that are relevant to implementing cloud-based IT systems. These focus areas are called *perspectives*. Each perspective is covered in a separate whitepaper. This whitepaper covers the Platform Perspective, which focuses on designing, implementing, and optimizing the architecture of the AWS technology that you use in your cloud adoption initiative.

Introduction

Your organization can use the AWS Cloud Adoption Framework (CAF) guidance to explore how different departments can work together on one or more cloud adoption initiative. Guidance is separated into the following focus areas, called *perspectives*: Business Perspective, Platform Perspective, Maturity Perspective, People Perspective, Process Perspective, Operations Perspective, and Security Perspective. The Platform Perspective components describe the structure and design of a cloud-based IT system, or a hybrid IT system that spans both cloud and non-cloud environments.

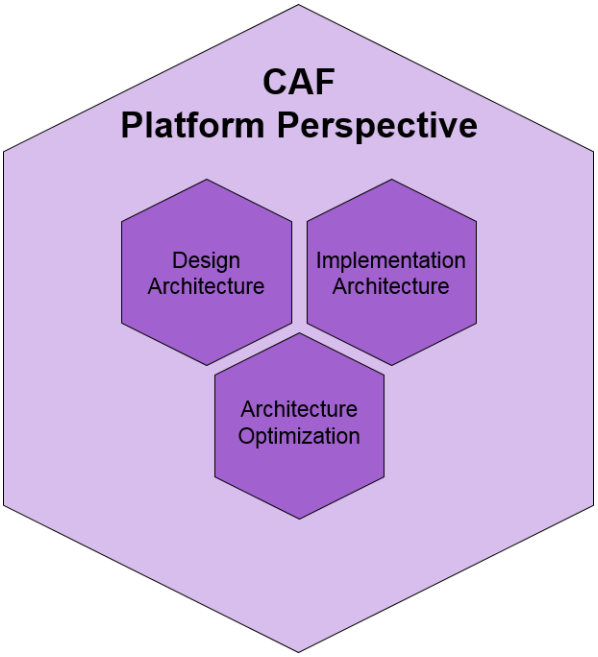

Figure 1 Components of the Platform Perspective

The rest of this whitepaper describes how the perspectives translate into activities that your organization can perform.

This whitepaper covers design architecture and implementation architecture. You can also benefit from principles and patterns for rapidly implementing or

experimenting with new solutions on the cloud, or migrating existing non-cloud solutions to the cloud, which will be covered as part of optimization.

Embracing Agility

Many organizations already use agile development to increase the velocity of their anticipated business outcomes. However, some businesses experience difficulty in achieving agility all the way through to deployment and operations. Consider embracing agility if you want to increase the velocity of achieving your anticipated business outcomes. For example, you could form a team to initiate a project and, with limited analysis, use AWS services to create a proof of concept (POC). If the POC is successful, you continue. If not, you select a different approach. The AWS platform creates a low barrier for experimentation, and allows you to rapidly deploy servers. When you complete your POC experiment you can shut down the AWS services environment, and no longer pay for resources. When your solution is ready for end users (the minimally viable product), you can gather the users' feedback and use it to inform priorities for future feature releases. By documenting the different phases of your cloud journey as it progresses, you can create a complete picture of the IT environment. Consider storing the artifacts that you create using incremental experimentation in the source code management system that you use today for storing and revising your application code.

You can complete the process of describing a business need and transitioning it into an IT solution using an iterative approach. In addition, you can use an iterative process to provide delivery teams enough detail so that what they build provides the intended outcome. Figure 2 illustrates how an IT capability maps to the services that deliver the capability.

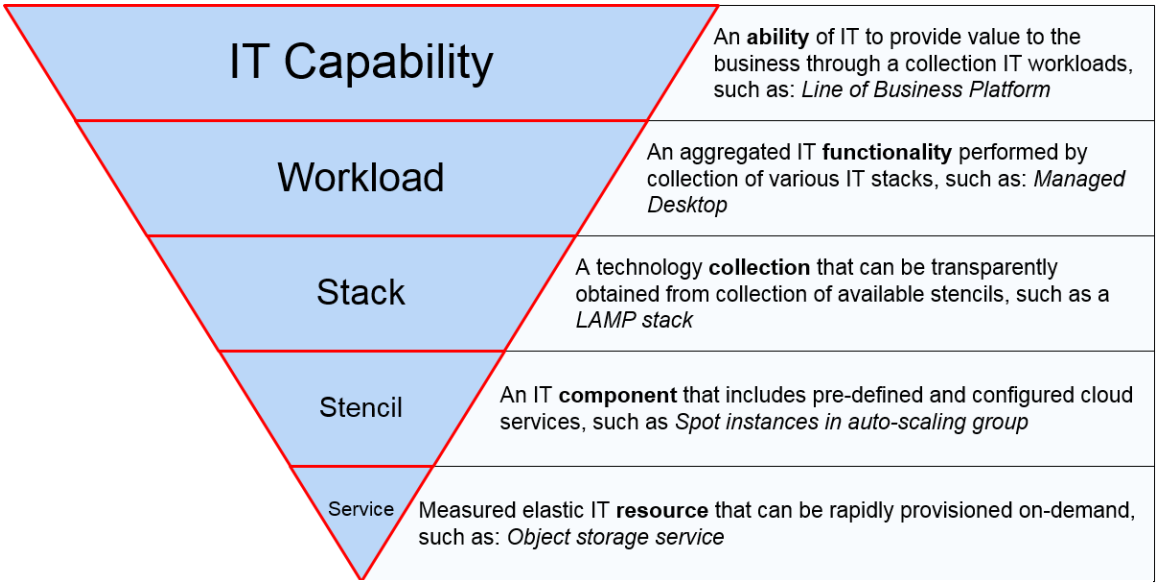

Figure 2: Example of Architectural Mapping from Capability to Service

When you use an iterative architectural approach, you can focus more time on business needs and goals. As business needs change and more information is surfaced, the technical architecture you use to deliver the business capability to the customer can shift to match the business need. You can also iterate faster, trying out new things to see if they work with minimal barrier to entry, due to utility pricing. The iterative approach makes it easier to roll back changes or stand up a parallel environment to test new features.

You can use a combination of AWS services to create IT capability, and use the AWS Service Catalog to centrally manage commonly deployed IT services. You can also use AWS services that provide a specific IT capability, such as Amazon Glacier for data archiving.

There are several components to consider from the Platform Perspective:

The Design Architecture component: Look at the common design patterns used in your implementations and identify common needs and redundancies.

The Implementation Architecture component: Look at the security, data handling and retention, log aggregation, monitoring needs, and common operational patterns.

The Architecture Optimization component: Identify your optimization strategies, what tools and processes need to be changed, and what automation can be used.

Design Architecture

The Design Architecture component of the Platform perspective promotes the engagement of stakeholders from many parts of the organization. In your cloud adoption scenario, you need to provide different views on your architecture to each stakeholder. For example, as you work with business sponsors to design a solution you can contextualize the architecture to describe how IT can be used to achieve the expected business outcome, and what the costs, returns, and risks might be.

Prior to an AWS adoption journey, your organization should consider modifying its governance and architectural principles to include AWS architectural principles. If you have not done so, then try using the iterative method described earlier to establish these principles. You can build methodologies and processes using sprints, just as you build applications. As you build, you can validate the design of your conceptual architectures against your governance and architectural principles.

Conceptual Architecture Activity

Conceptual views are technically abstract, but they should be described in a context that is familiar to business users. Use the conceptual architecture to define the business context of an IT system with business models. This is where you balance short-, medium-, and long-term business goals and concerns for IT initiatives.

Three key components of a conceptual architecture are business vision, goals, and objectives. Use the conceptual architecture to understand which capabilities will be needed as part of the logical or functional architecture that will describe the solution. Figure 3 illustrates an example conceptual architecture that describes where AWS services are applicable.

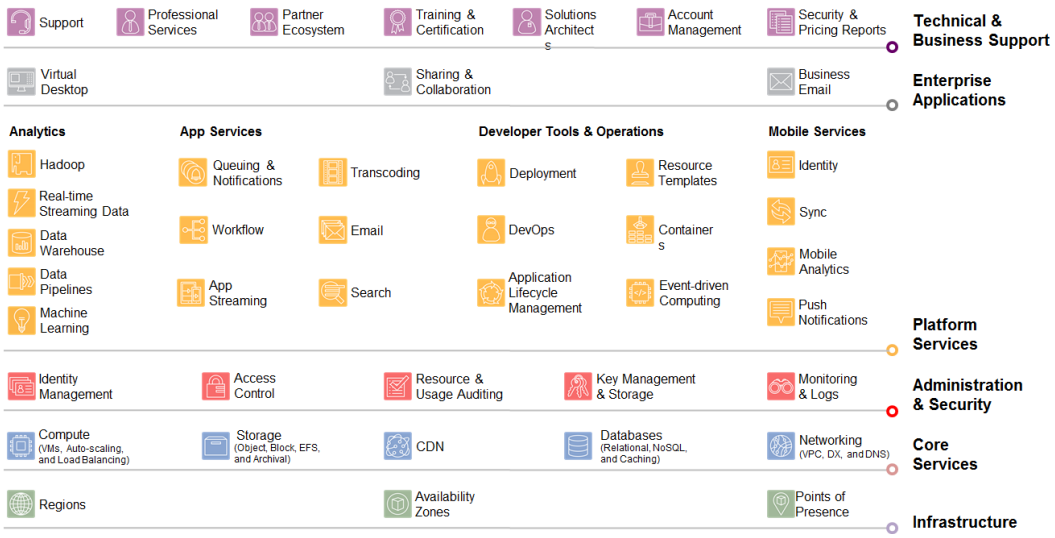

Figure 3 Example of a Conceptual Architecture

Using AWS, the creation of a conceptual architecture can become more iterative. You can use AWS services as part of the development effort by using experimentation to validate and evolve the approach. As business capability concepts are proven, development teams can start work on delivering functions and features into production. With quicker delivery, end-user feedback can be used to verify whether business objectives and compliance requirements are being met with the current technical approach.

Implement automated testing to test your rapidly iterating conceptual architecture. This not only minimizes the introduction of bugs into your application, but also includes continuous compliance as part of continuous delivery, helping to ensure that changes to your application do not affect your organization's security posture.

Logical Architecture Activity

Logical (or functional) architectural views describe the building blocks of the IT system and their relationships without getting into the technical details of how the functionality is implemented. The logical architecture contains the data flow and capability models that relate to the business models that meet the business outcomes.

Quality attributes, dependency mapping, and plans for obsolescence can be identified, documented, and addressed as part of designing the logical architecture. A logical architecture (Figure 4) that uses AWS can make use of geographical duplication as well as the elastic nature of AWS services. Using design principles that take advantage of these characteristics will allow system capacities to expand and shrink as loads expand and contract.

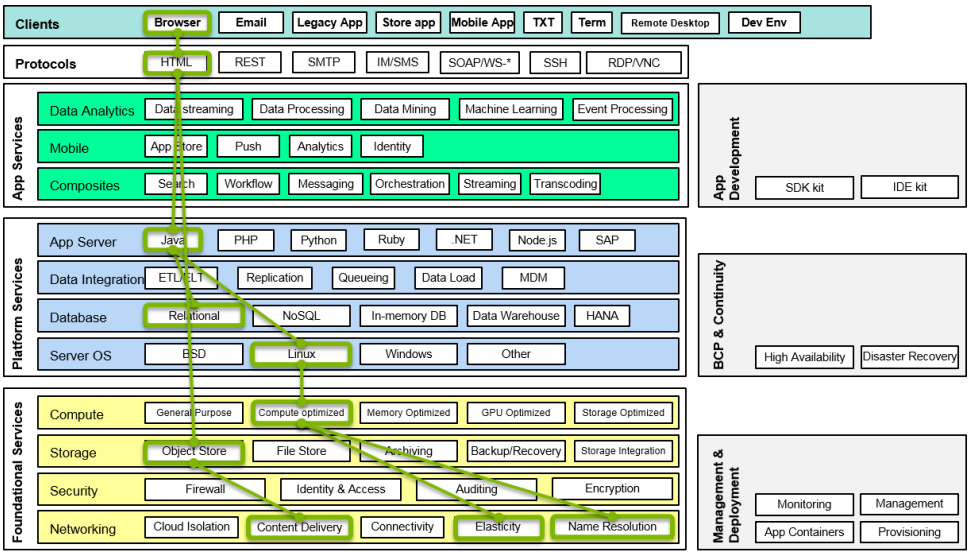

Figure 4 Example of a Logical Architecture Diagram

You can use different approaches based on the type of project your organization is designing. Projects with a long duration typically are used in predictable, repeatable environments or environments where refinement of approach is not possible or recommended after decisions are made. These types of initiatives are driven with top-down control over outcome. An example of such an initiative is shutting down a corporate data center after a decision to move to the cloud.

Initiatives with a short duration are driven with bottom-up freedom over outcome. Change in direction is expected and may be encouraged for better alignment with shifting business needs.

There are also hybrid approaches to initiatives where the goal is to migrate and decompose a monolithic mission-critical solution or environment. These initiatives will combine the best aspects of heavy up-front planning with the freedom to innovate as needed to deliver optimized customer outcomes.

Considerations

- **Do** use feedback from delivered features to review and revise the conceptual architecture with the business team.
- **Do** minimize the number of architectural principles to allow the greatest flexibility in solution development.
- **Do** stay focused on customer outcomes and business objectives rather than technical solutions.
- **Do** experiment with AWS services to experience, learn, and prove that your logical architecture will achieve the desired business outcome.
- **Do** focus on short duration project scoping and iterative processes for systems of interaction where outcomes are more fluid.
- **Do** consider the practice of creating logical architecture as a dynamic process.
- **Do** limit the amount of redundant technologies to prevent "technology sprawl" and allow for focus and specialization.
- **Do not** make functional and implementation architectures dependent on a complete conceptual architecture. Consider identifying a key objective and starting design and delivery of that functionality. Use the feedback from adoption of the features as input in the evolution of the conceptual architecture.
- **Do not** attempt to create the perfect architecture up front. Consider starting with the highest risk/reward scenario and use experimentation to prove your approach.

Implementation Architecture

The Implementation Architecture component of the AWS CAF Platform Perspective describes the detailed designs within the IT system and the specific implementation components and their relationships. This architecture also defines the implementation of the system's building blocks by software or hardware elements.

The implementation architecture for an AWS environment describes the design of the technical environment. The description is broken into layers, with each layer providing information for a specific team in the organization. AWS reference architectures are available at http://aws.amazon.com/architecture.

Figure 5 illustrates a high-level implementation architecture. This artifact works best online, where you can enable clicking on each item for more information, and you can plan for automatic updates.

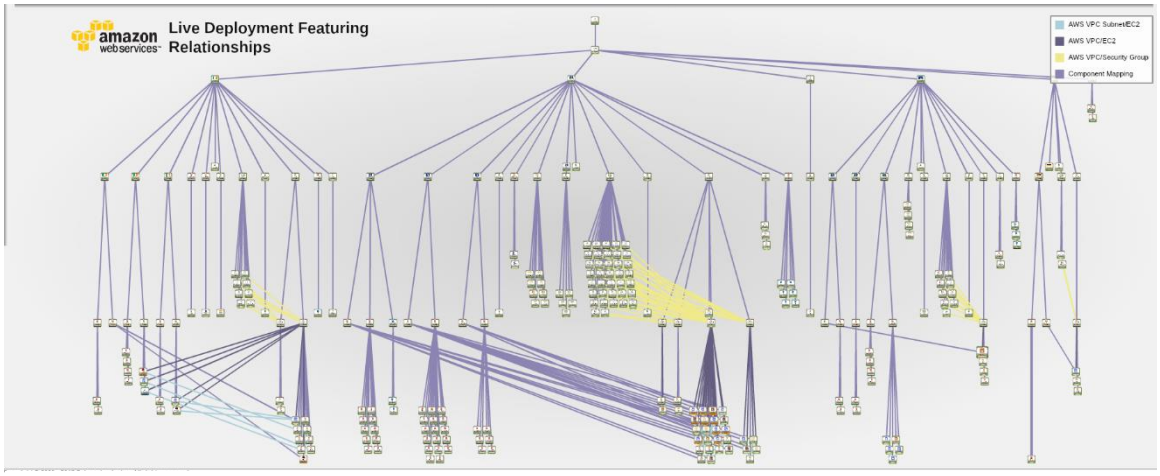

Figure 5 Example of an Implementation Architecture

Describing the AWS environment and providing guidance on usage will be a critical portion of the implementation architecture development. Describing how resources, accounts, and tagging work, and the how the Amazon Virtual Private Cloud (VPC) environment is configured provides information that will help the organization determine which resources are consumed by various systems, applications, and initiatives.

The Information Architecture should set strategies for deployment, monitoring, auditing, and logging that will give you exposure to near real-time data. Set

security, data retention, gateway, and routing strategies and policies so your delivery teams have the information they need to enable control over the AWS environment as it grows.

Include taxonomy and naming conventions as part of the metrics, monitoring, and chargeback implementation. The actual running environment will change continuously and will be best viewed through dashboards with near real-time information.

Dashboard information can be represented graphically or by using lists. If you use a graphical dashboard, users could click the graphic to show additional detail. If you use a list in your dashboard, users familiar with spreadsheets can find information in well-defined columns. Figure 6 shows a graphical dashboard that can provide near real-time information.

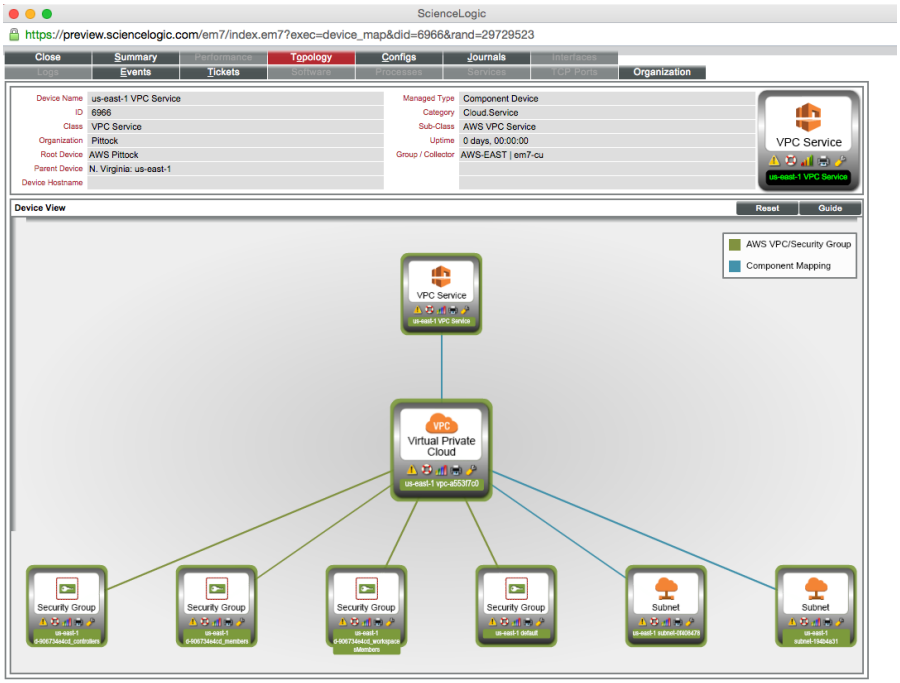

Figure 6 Example of a Graphic-based Near Real-time Dashboard

Consider prescribing a taxonomy and naming convention in the implementation architecture. Then you can implement this taxonomy as a tagging standard on AWS resources. To increase confidence and reduce risk, you can leverage the AWS environment during implementation architecture creation. When you use AWS, the environment can be created and tested for verification or certification earlier in the release cycle. Additionally, tools are available through AWS and the

AWS Marketplace that can automate processes and shorten the time needed to deliver, test, and operate AWS-based environments.

Defining an operational playbook for how you are going to deploy and operate your systems will help ensure consistency and repeatability of success. This playbook should also be iterative in nature, with the constructive feedback implemented in systems that did not have this capacity at the time of creation.

Considerations

- **Do** identify a network connectivity strategy for AWS services.
- **Do** outline AWS components to be used (services/features).
- **Do** define security controls (native vs. third-party tools). Greater details are available in the *AWS CAF Security Perspective* whitepaper.
- **Do** define data security and retention policies (encryption, backups, snapshots, third-party tools).
- **Do** create and work toward an automated deployment process to reduce the impact of human error and introduce portability.
- **Do** create an operational playbook. More information on this topic is available in the *AWS CAF Operations Perspective* whitepaper.
- **Do** outline a monitoring strategy.
- **Do** outline a logging strategy that validates that your logging system can manage the amount of information you decide to collect.
- **Do** create a strategy for resource tracking as part of your implementation architecture, ensuring that resources are appropriately tagged at the time of deployment. This can also be extended into cost allocation tagging.
- **Do not** let application environments form in an ad hoc fashion. Choose a strategy to organize your application environments.

Architecture Optimization

The Architecture Optimization component of the AWS CAF Platform perspective promotes the adaptability of an architecture that uses AWS—as business needs change and as new and better technical solutions become available, your architectural decisions can be modified and adjusted. Since physical computers are not purchased, the long lead-time for procurement, staging, burn-in, and configuration is no longer necessary. Because you can continue to optimize your architecture during the design phase, this process can completed with less up-front information; your decisions can change and be implemented as needed.

As you adopt AWS services, a key focus should be on building tacit knowledge in the organization. Creating a centralized repository with principles, patterns, best practices, a glossary, and reference architectures will help ensure the rapid growth of AWS skills in the organization. As you start an automated and agile deployment process, the centralized information repository allows systems and people who deploy applications to access the governing principles as well as the pieces and parts that they own.

Cloud Design Principles and Patterns Activity

Adherence to the software design principles and patterns that you document will improve quality and productivity and reduce risk during solution development. All delivery teams can follow these principles when designing and building solutions. A *pattern* is a proven approach to achieving a result. You can automate patterns that you use frequently to improve efficiency, consistency, reliability, and supportability. Consider following these best practices:

- **Provide guidance that captures reusable approaches**, leverages an infrastructure as code approach, and treats that code like application code (source control, code reviews, etc.).
- **Create a baseline of language and understanding** across the technical organization to ease communications. This might include creating a taxonomy and a dictionary or a glossary describing how things will be named and organized.
- **Educate everyone to a foundational level** to provide common language and understanding. Building fluency in the language of AWS cloud adoption and explaining the taxonomy and naming conventions will help accelerate

familiarity with and ability to use cloud-based technologies and approaches across the organization.

- **Use fast track or factory principles to create common approaches** with reliable results. Provide documentation that describes diagrams, naming conventions, code review guidance, and so on to provide a common language, approach, and expectations. Using wiki-based tools for documentation will allow teams to update documentation and keep it current, and will provide a single authoritative source for guidance.
- **Create** a governance process and/or team that ensures and/or audits the outcome of patterns and intended results.
- **Provide an "Andon cord"** for the deployment team to use if they see something that doesn't fit in with their understanding of patterns.

Application Migration Patterns Activity

Proven approaches for migrating IT systems to the cloud are available as migration patterns.

Consider organizing applications in a way that helps identify and introduce patterns that you can use with predictable results. Two of the more commonly used pivots are business criticality and data classification. Understanding which categories of data are associated with which applications will provide valuable insight. Another useful pivot is level of mission criticality. Depending on your needs, you could also consider organizing by systems of record versus systems of interaction, monolithic applications versus highly decomposed applications, or new applications versus applications near the end of life.

One approach that you can take is to organize your applications into five groups based on the action you want to plan for each application. The different actions include retiring, retaining, replacing, re-hosting, refactoring, and rewriting. Figure 7 illustrates this five-group application migration pattern.

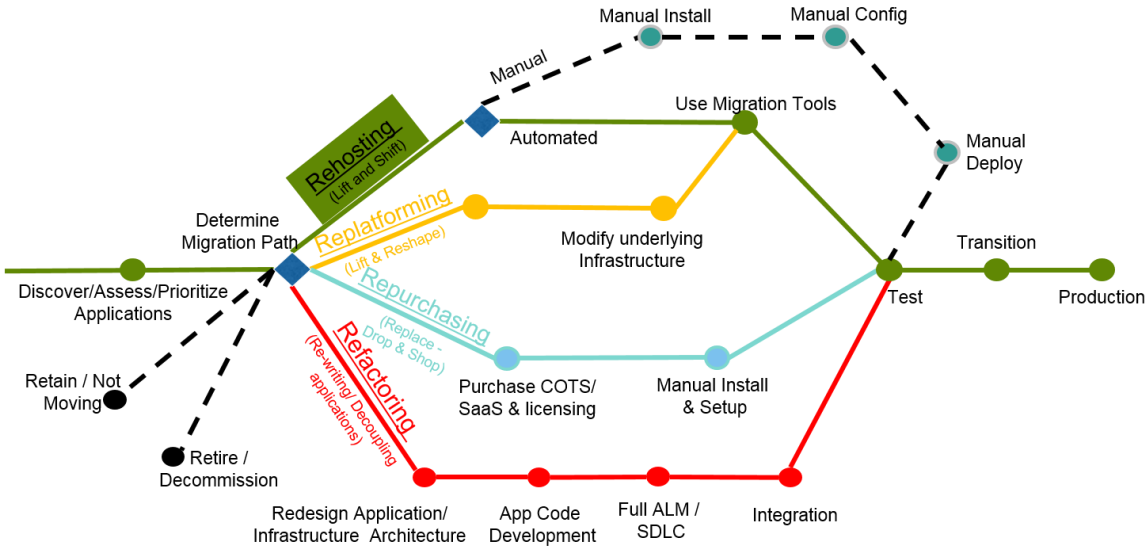

Figure 7 Graphic Representation of an Application Migration Pattern

You can also use an inventory of current data center applications and their dependencies to determine which applications to migrate and when. This could potentially allow you to avoid a costly equipment refresh, pushing away from capital expenditure (CapEx), and taking advantage of AWS utility pricing.

For making decisions about which patterns to leverage, consider creating a Center of Excellence (CoE) team to select patterns that enable the shortest time to value. Another approach is to organize and prioritize by ease of effort to migrate. For example, you could decide to migrate development and test applications first, followed by self-contained applications, customer training sites, pre-sale demo portals, and trial applications. During the migration, consider prioritizing a Tier 1 application to gain visibility and endorsement from executive sponsors.

Consider developing new applications or refactoring existing applications in the AWS environment. For existing applications, you could migrate applications to the AWS cloud environment and prioritize rework or optimization initiatives. The refactoring can be enabled by the agility of deployment on AWS.

Considerations

- **Do** consider new applications for migration first.
- **Do** start development of new capabilities or rewrites of existing capabilities in the AWS environment.
- **Do** take advantage of capacity concerns as a reason to prioritize development in the cloud.
- **Do** consider using code review (both for application and infrastructure code) to provide a feedback loop that improves process and reduces technical debt.
- **Do** consider using wikis to provide access to guidance that can be updated and maintained over time.
- **Do** leverage AWS cloud adoption as a way to fast track maturity of combined roles and skills thinking. This would manifest as a developer/security/operations mindset and coding architectural models to validate approach.
- **Do** use AWS cloud adoption to institutionalize a scalable, service-oriented architecture (SOA) approach to separate concerns and to enable integration of reusable services and limit the amount of code maintained.
- **Do** create patterns that assume failure by building in recovery code with features such as circuit breaker patterns, caching, and queuing, and exponential back-off.
- **Do** write code with an eye towards reuse through exposed API endpoints for easy discovery, integration, and reuse.
- **Do** introduce your deployment team to your development team. Empower both teams to fully appreciate the benefits of scalable infrastructure and utility pricing.
- **Do not** optimize a solution before it is well architected.
- **Do not** start migrations without operational processes defined. Consider defining backup and recovery guidance as an initial step in a migration effort.
- **Do not** manually migrate all applications. Consider using automation to scale and accelerate migration of applications (migration factory).
- **Do not** wait to automate something. If you're deploying the same thing twice manually, invest the time in automation.

CAF Taxonomy and Terms

AWS created the Cloud Adoption Framework (CAF) to capture guidance and best practices from previous customer engagements. An AWS CAF *perspective* represents an area of focus relevant to implementing cloud-based IT systems in organizations. For example, when a cloud solution is to be implemented, the Platform perspective provides guidance on designing, implementing, and optimizing the architecture of the AWS technology that you plan to use in your cloud adoption initiative.

Each CAF perspective is made up of components and activities. A *component* is a sub-area of a perspective that represents a specific aspect that needs attention. This whitepaper explores the components of the Platform perspective. Within each component, an *activity* provides prescriptive guidance for creating actionable plans an organization can use to move to the cloud and to operate cloud-based solutions.

For example, *Design Architecture* is one component of the Platform Perspective, and creating logical architectural views that describe the building blocks of the IT system and their relationships may be an activity within that component.

When combined, the AWS Cloud Adoption Framework (CAF) and the Cloud Adoption Methodology (CAM) can be used as guidance during your journey to the AWS cloud.

Conclusion

Translating business outcomes into technical solutions is still a necessary step in the IT lifecycle. By adopting AWS services, you have the flexibility to change an architectural decision after more information is gathered and as assumptions are tested and technology advances. The Platform Perspective provides an approach to separating a complex set of ideas and decisions into manageable components.

Use the design component to facilitate discussions with business stakeholders and provide an abstract level of detail to describe how business outcomes will be accomplished.

Use the implementation component to facilitate discussions with technical teams who are responsible for creating, delivering, and maintaining solutions at a level agreed upon with the business stakeholders.

Use the architecture optimization component for approaches and patterns that provide predictable and repeatable results. For example, when you use an application migration pattern you can organize and categorize groups of applications and follow a common approach to migrating to an AWS environment. You can also create a small set of principles that all technical team members can use to help with key decisions. This ensures that a common approach to making decisions is used across the organization.

Notes

[1] https://do.awsstatic.com/whitepapers/aws_cloud_adoption_framework.pdf