Cloud-Driven Enterprise Transformation on AWS

AWS Whitepaper

Cloud-Driven Enterprise Transformation on AWS: AWS Whitepaper

Table of Contents

Cloud-Driven Enterprise Transformation on AWS

Publication date: **May 11, 2021**

Abstract

Enterprise customers often request advice and learnings from Amazon Web Services (AWS) on how to accelerate the velocity of their cloud transformation. This document describes an integrated approach to achieve speed and accelerate time to value while maintaining control and corporate guardrails. This approach enables teams, delegates decision rights, promotes experimentation, and leverages a continuous improvement mindset. The approach provides IT and business leaders with a mechanism to tailor and scale their value-driven cloud transformation programs.

Introduction

This whitepaper describes an integrated approach to accelerating the velocity of cloud transformation based on agile principles to couple speed with control and accelerate time to value, while empowering decision rights, experimentation, and continuous improvement across teams.

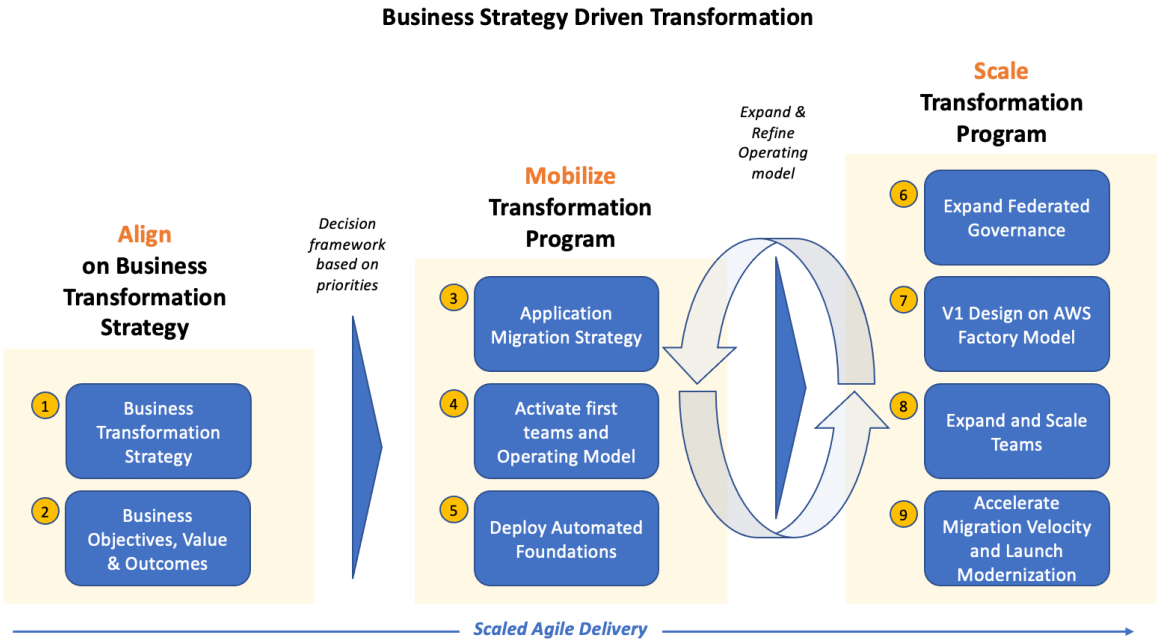
Business transformation objectives can be met with the following five levers:

- Expand business / IT partnership with clear executive ownership
- Initiate a multi-layered cloud migration strategy that accelerates simple migrations, begin tackling complex application design on AWS, and identify workload dependencies
- Implement a consistent enterprise AWS design decision support process
- Define and launch a “factory model” to accelerate migrations to AWS, focusing on activating delivery teams to move their business applications to AWS
- Scale out the cloud transformation program and expand foundational capabilities, automation, and re-usable assets based on prioritization of highest value and re-use

Successful transformation programs are business-driven. They have clear governance of key milestone objectives and success metrics coupled with a people strategy that accommodates a variety of organizational / team levels of AWS expertise. Enterprise transformation is supported by adopting an agile approach with joint decision-making between business owners and IT teams on the timing and degree of transformation. In every case, the degree of business involvement and transformation needs to be balanced with other competing delivery priorities and a clear business case for change.

Successful cloud transformation programs enable teams to work and collaborate using agile principles to improve collaboration and joint prioritization. An agile approach breaks down large tasks (such as enterprise migration) into manageable, iterative pieces of work delivered and measured incrementally. This approach also provides a feedback mechanism to measure team velocity, adapt approaches to increase performance, and identify and optimize dependencies on teams that might not use the same model.

The AWS Cloud Adoption Framework provides further details on the pillars of successful cloud adoption.



Business strategy-driven transformation approach

Positioning your move to AWS as a business transformation project

Traditional IT organizations operate with a boundary between business, application, and infrastructure teams. IT executives often ask AWS how to accelerate their AWS migration "with little to no business or application developer involvement" (see Table 1). This is not an effective strategy, because even in the simplest scenario application, User Acceptance Testing (UAT) resources need to prioritize the migration effort and sequence cutover along with planned application releases.

Migrating internally developed business applications to AWS, even in a "like for like" architecture approach with minimal optimization or refactoring, still requires a thoughtful "full stack" design. This entails addressing all the design elements, including:

- Infrastructure design
- Application architecture
- Testing suite
- Deployment processes
- Resiliency considerations
- Operational capabilities

Even with an architecture on AWS similar to the on-premises configuration, it is the application development team who will own the testing and the evolution of the system on AWS. In some cases, such as data center exits with a compelling date driven by an urgent divestiture process or significant financial penalties for missing lease end dates, a broad "forklift" automated migration path for a majority of assets may be appropriate. This requires coordination and prioritization with networking, security, applications, and business stakeholders. Companies that do not engage all aspects of their organization to accelerate through the most complex phase of dual operations (on-premises and on AWS) often stall early in the transition, diminish the business case, and miss the opportunity to capture the value of teams working "all in" and innovating on AWS.

An integrated teaming approach to a business-strategy-driven cloud transformation requires involvement of key business and technical stakeholders committed to AWS migration. Speed to value is achieved by driving incremental optimization to energize the benefits of tighter collaboration that is lacking in traditional silos of legacy IT governance models. The choice is not between moving applications to the cloud "as is" or full refactoring to enable cloud-native capabilities. There is a spectrum of options in-between (described in the next chapter of this document). Companies who move fastest empower teams to evaluate trade-offs, document their decisions, move quickly, and use data to measure performance and adjust.

Table 1 – Traditional vs. modern view of IT transformations

Traditional view	**Modern view**
Think of migration to cloud as an infrastructure project • Driven by IT hosting services teams • Data center infrastructure lens • Limited business involvement	Adopt business application / business value view • Use migration to align IT portfolio to business priorities

Traditional view	Modern view
	• Use migration to accelerate digital transformation and increase productivity of builders • Include factors of speed, resilience, cost, and agility in migration strategy • Automate and eliminate non-differentiated heavy lifting • Focus on business applications in production on AWS

Based on observations of the authors with large enterprise transformation programs, large enterprises often stall at the cloud migration point of 10-20% of their business application portfolios. Based on AWS experience, cloud business transformation typically stalls for three primary reasons:

- No clear alignment on cloud transformation objectives across full senior team of business leadership, including the CEO and direct reports, as well as management board alignment on priorities to meet business strategic objectives
- A lack of clear, aggressive, date-specific top-down business-driven goals
- No long-term plan to support business goals with measurable milestones and clear ownership. Without such a plan it will not be possible to track and drive success.

To accelerate a successful program, increased business alignment and partnership is necessary, manifested in the form of a business transformation strategy. Sometimes the disconnect in leadership alignment is generated by a limited understanding on the part of business leadership of the cloud value proposition. Educating business leaders across the entire organization on their role in successful cloud-driven transformation is essential to the speed and resolution of competing priorities.

Companies with broad business commitment move faster through the “dual operations” hybrid phase of operations, unlocking the full value sooner when departments and portfolio teams are “all in” on AWS.

Expanded business partnership

For best results, do the following:

- Identify a C-level executive sponsor to help drive your transformation program and set priorities in context of other critical business objectives.
- Communicate value proposition broadly to business and enlist a set of executive leaders to endorse the program, own their business line’s completion of the migration program, and commit the necessary resources (developers, UAT testers, executive sponsors).
- Create strong joint ownership of program outcomes.
- Set clear corporate goals.
- Tie program to business enablement.
- Measure progress and outcomes.
- Incentivize leaders to meet or exceed their timeline and modernization goals.

Executive sponsors can support culture change by communicating and supporting communication and failures along the way. Problems will be encountered; this is natural. With well-planned and well-tested deployments on AWS, most cutovers are non-events. When things do go wrong, failback procedures guide teams to quickly recover, adjust, and try again. An evaluation and correction of errors helps ensure

that avoidable mistakes are not repeated. Senior leadership team support for experimentation and controlled failure is essential to gain speed and unlock value.

Developing a business-driven cloud strategy

Working with customers, AWS sees two main factors inhibiting progress on a tactical level:

- Technology teams are challenged when making decisions on the AWS architecture to find a clear path forward that balances modernization and migration progress.
- A lack of clear criteria to balance between the effort required to achieve the platform transformation and the need to continue to deliver new business capabilities (such as feature releases) on existing systems.

Engineers and architects often see the move to AWS as an opportunity to deliver on aspirational projects and implement multiple major changes. Moving to project mode with dedicated team resources from business and IT teams will accelerate change, but do not refactor everywhere. Successful companies take an approach that balances speed, addresses technical debt where it inhibits modernization, and defers lower priority optimization until after deployment on AWS.

Always map technical debt to business value by prioritizing effort along strategic modernization themes aligned to business objectives (such as time to market) or strategic initiatives (such as near real-time data availability, next best action, customer 360, and personalization).

Table 2 – Traditional vs. modern view of IT migration strategy

Traditional view	Modern view
Try to develop a cloud strategy based only on asset data and technical considerations. • Server / VM-based strategy • Start with easy / cloud-compatible applications • Focus on deployments in development environment • Leave challenging problems for next year	Business requirements drive cloud strategy and migration path. • Modernize what matters • Prioritize degree of change based on business value (move to managed, cloud native, microservices) • Empower builders with secure self-service • Address technical debt and End of Serviceable Life (EOL) across the application “full stack” to support business capability gaps, security, and resiliency requirements • Start design, planning, and build for toughest migrations while migrating simpler use cases to accelerate experience • Business applications in production on AWS as a primary program health KPI

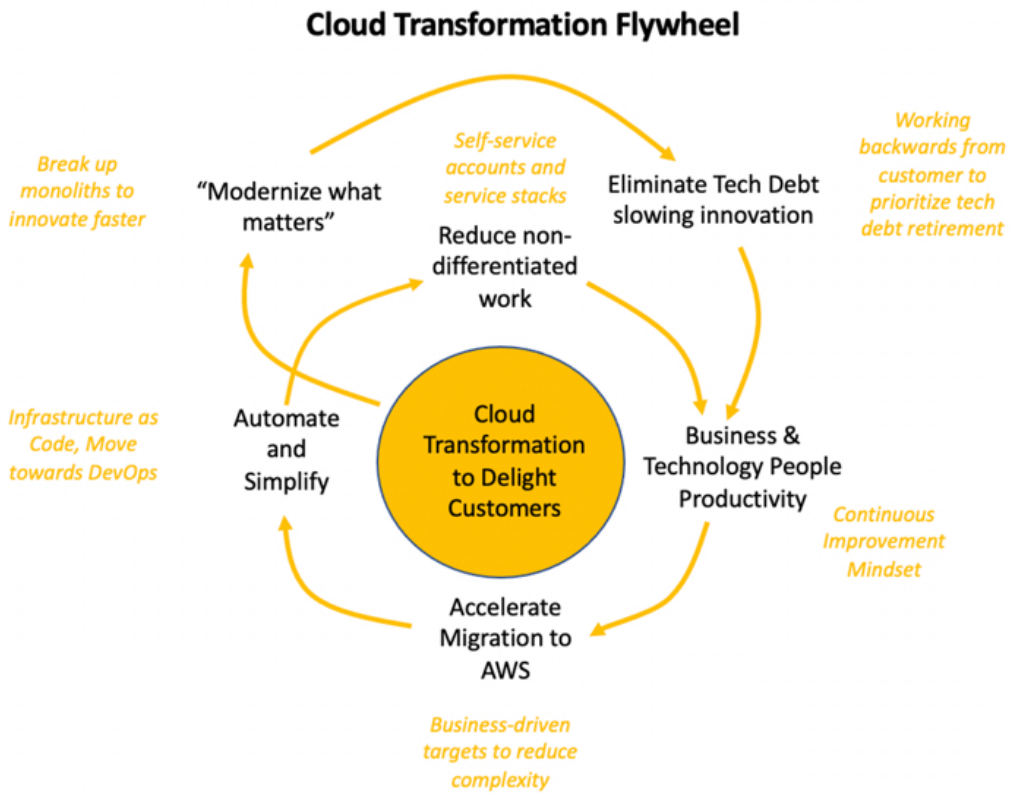
It matters how your business processes are supported by your application portfolio. Your business priorities should matter too, as well as your transformation timeline. Along with the 6 Rs (six common migration strategies for moving applications to the cloud), three cloud migration strategies emerge:

- **Move, then optimize (move quickly, then continue optimizing where there is business value)** — Move quickly with traditional lift and shift to capitalize on immediate run cost savings, then optimize in the cloud. Identify longer term modernization and refactoring opportunities during design review, and place them in the backlog to address after migrating to AWS. In some cases, systems that are

planned for run-out in 3-5 years might benefit from migration to a co-location facility adjacent to an AWS Region, so that dependent, cloud-ready workloads can move before rewriting systems to resolve dependencies.

- **Modernize what matters (larger refactoring, such as the mainframe where there is business value)** — If priorities and resources are aligned, invest in incremental or full modernization depending on the business value, technical debt, and other priorities. Begin evaluation of opportunities to resolve structural dependencies on legacy workloads, including high technical debt systems and common mainframe and mid-range systems. Retire, re-purchase, retain, or refactor these legacy workloads based on business need.
- **Minimal optimization to fix problems stopping you from moving faster** —Upgrade or renew outdated technologies across the full stack, and decouple legacy systems to address pain points that slow down development of new capabilities or drag resources into non-differentiated work. In most cases, this also require close coordination with application, testing teams, and business product owners to determine how far to modernize before migrating to AWS.

Navigating those three strategies across your business applications portfolio will enable the flywheel for your cloud transformation.



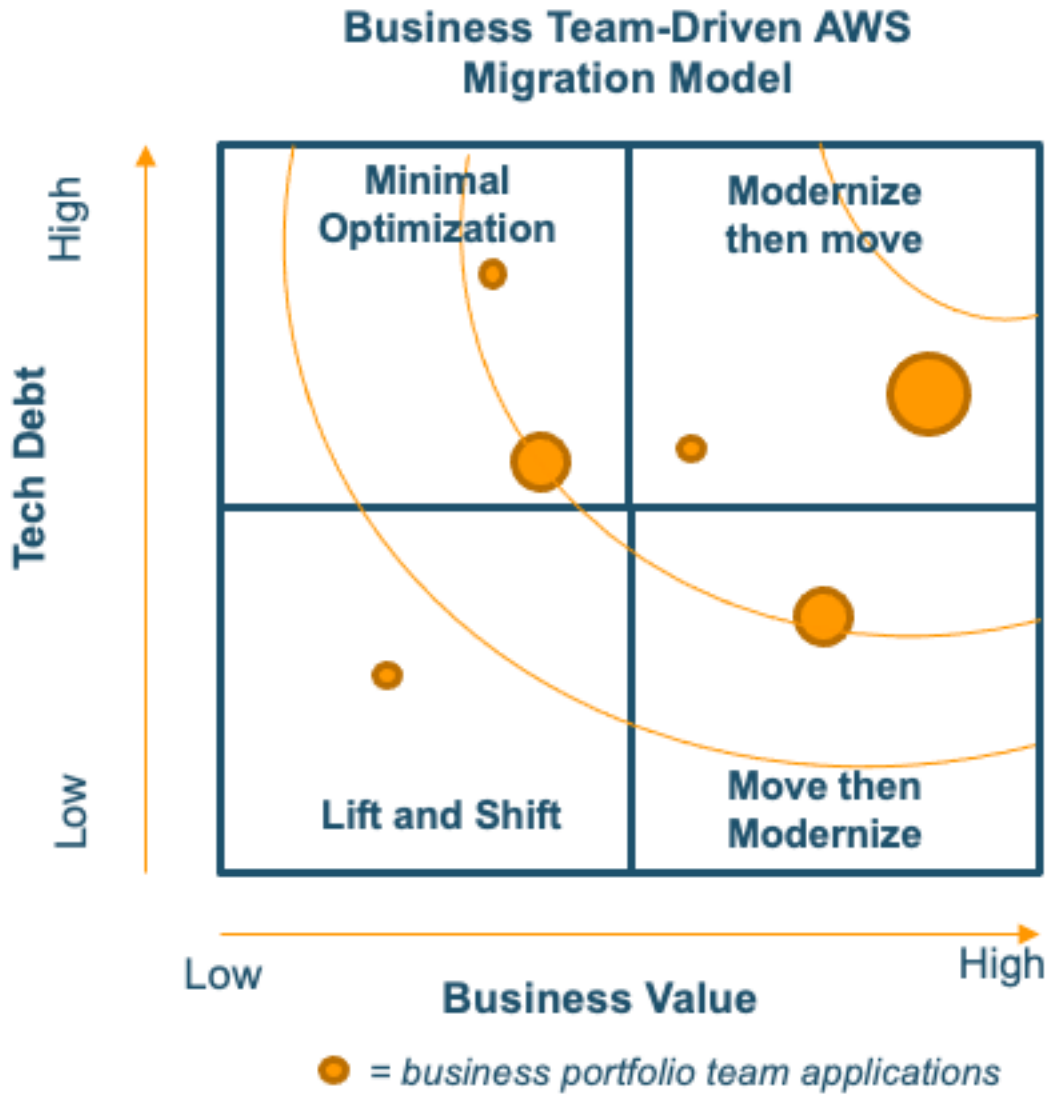
Cloud transformation flywheel model

Customers often think about modernization as an "all or nothing" move to cloud-native architecture. There are so many variations between a "lift and shift" and full-stack refactoring. When you adopt a mindset that you are not just planning to survive four to five years of on-premises infrastructure lifecycle, you have options to put optimization ideas in your system backlog and prioritize improvements after going live on AWS.

With so many options, analysis paralysis can set in. Tackle the change volume that your teams can manage, along with other delivery priorities. Tackle the modernization projects that unlock the

most value. Empower teams to make these trade-off decisions, and encourage decision-making and movement over perfection. The most important thing is to make decisions efficiently, understanding that delaying decisions is a decision to do nothing.

Evaluating options that are not realistic based on team skills, resources, or other priorities get a lot of companies stuck in analysis paralysis. Embracing a mindset of data-driven, continuous improvement permits teams to focus on incremental change, smaller steps, and using performance data and feedback to inform the ongoing path of modernization and optimization.



Application portfolio rationalization decision model

On AWS, a number of quick paths to modernization can be adopted, such as:

- **Infrastructure as code** – Automate provisioning of infrastructure using templates, quick starts, and secure stacks vended through the AWS Service Catalog.
- **Incremental deployment automation** – Move toward automated deployments for middleware and application code, leveraging a common cloud-based code pipeline with controls built in and adoption of DevSecOps practices.

- **Auto-scaling** – Allowing horizontal scaling, instance swaps, and self-healing / auto-recovery mechanisms to automate manual operations tasks.
- **Enhanced resiliency** – Standard pipeline to enable redeployment / failover to another Availability Zone (AZ) or Region for business-critical applications with resiliency / disaster recovery requirement.
- **Move to managed services** to reduce undifferentiated heavy lifting such as managed databases (Amazon Relational Database Service (Amazon RDS) and so on).
- **Move to containers** – Begin to break down monoliths into smaller services and containerize applications for ease of management and deployment, starting with components for which Docker images are ready off the shelf.
- **Opportunistic refactoring** – Rewrite particular pieces of your application stack; for example, by moving event-driven components to serverless (such as AWS Lambda, AWS Step Functions, and AWS Batch), or your proprietary database engine to AWS-native data stores (such as Amazon Aurora).
- **Standardize application layers** – standardize application stack and architecture; for example, consolidate technologies for web / app / middleware layers where minimum refactoring is sufficient (such as moving to an Apache Tomcat shared layer).
- **Repurchase** – For example, move to Software as a Service (SaaS) or purchase a replacement from the AWS Marketplace for cloud-based commercial off the shelf (COTS) applications.

Using this approach, a number of common patterns and paths emerge, as shown in Table 3:

Table 3 – Application deployment patterns

Select the pattern and the path (mode of migration) based on relative business value, rate of change, team skills, and capacity.

Pattern	Automated migration	Pattern-based migration	Bespoke / custom architecture	Incremental automation and resiliency		Average % of mix
	Includes VMware Cloud on AWS or CloudEndure	Containers or pattern build, review incremental changes if required, leverage pre-built quick-starts	Custom architecture	During move (pipeline deployment)	Move, then optimize	
Refactor (re-write, decouple, move to managed, or cloud-native)		**X**	**X**	**X**		~5-10%
Repurchase / Replace (drop and shop)	Move to SaaS or purchase a replacement from AWS Marketplace and migrate users to new platform.					<5%
Replatform (shift and reshape)		**X**	**X**	**X**		~40%

Pattern	Automated migration	Pattern-based migration	Bespoke / custom architecture	Incremental automation and resiliency		Average % of mix
Rehost (lift and shift)	**X**	**X**		**X**	**X**	~40%
Relocate	**X**				**X**	<5%
Plus: Retain on premises, forklift from colocation, or Retire						~5-10%

There are trade-offs for each path. Automated, tool-based migration is a replica of on-premises configuration, and will not improve your capability for continuous deployment or resolve any technical debt. Automated rehosting is the fastest path, and might be particularly useful if there is a divestiture or requirement for avoiding impact on existing vendor support arrangements. This can also be a path for rapid migration for time-sensitive data center exits, but it will still require close coordination with business stakeholders for testing and UAT, as well as a complete understanding of dependencies between applications.

The pattern-based approach for “like for like” migrations can help teams move at scale, re-use common design patterns, and add in incremental resiliency and DevOps capabilities with limited change to the application.

A custom migration would apply to complex, multi-tier, business critical applications (“big rocks”) requiring bespoke architecture. Based on the business case and competing priorities for time and resources, this path could still range from incremental changes to full modernization or re-write as required. This custom category includes large-scale databases and application re-writes that also might require refactoring of business workloads or business processes.

Adopting a consistent design decision approach

An important attribute of AWS is the flexibility to permit an evolving design, and the elasticity of components as business needs evolve or traffic demands vary from historical data or expected forecasts.

Architecture on AWS does not need to be perfect. An important mental model shift is required from the legacy on-premises world, where workloads need to be forecast for three to five years plus a safety buffer for hardware sizing, and the architecture must be "future-proof" to support unforeseen changes and requirements. On AWS this is no longer the case.

V1 architecture and hardware performance must meet your current requirements with a safety factor. Teams can architect auto-scaling capability into V1 to burst capacity when required, but even where auto-scaling is not supported by the current application architecture, server capacity can be scaled vertically, on the fly, with minimal to no downtime to meet unexpected traffic demands. (See the **V1 AWS design** (p. 11) section of this document.)

When in doubt, teams can over-size hardware, allow systems to run for a couple of months on AWS, then down-size or optimize the configuration and service costs after real-world performance on AWS is determined with data. For all of these reasons, teams should invest time in a V1 AWS design that provides the right level of incremental improvements based on business need.

Table 4 – Traditional vs. modern views of IT architecture decisions

Traditional view	Modern view
Architecture as one-way door • Design for 5-year growth forecast ('one and done') • Over-build to anticipate/guess future use patterns • Led by infrastructure: business not involved • Tool and technology focus	Modernize what matters—'V1 AWS design' architecture approach • Architect for cost, performance and operations • "Two-pizza teams" empowered to make decisions • Based on business need, quickly determine first step • Optimization backlog on V1 manifests design evolution over time

V1 AWS design

Design decisions regarding the degree of transformation in application and infrastructure architecture, configuration, and the degree of incremental deployment automation manifest themselves in the "V1 AWS design" implemented with the production cutover to AWS. V1 AWS design is the first architecture design on AWS that represents the first, best step to the cloud. V1 AWS design can range from "like for like" to full modernization or cloud native re-factoring, depending on business value and alternate uses of time and resources.

Cloud architecture decisions are best driven with the application owners, developers, and business partners. A good V1 AWS design document paints a clear path forward for V1 deployment. This V1 design includes trade-offs to determine what components of operating system (OS), database,

application code, resiliency architecture, deployment automation, and services architecture enhancements should be implemented with the migration to AWS (and incorporated in V1 design) or be deferred to the backlog for post-migration to AWS optimization.

A move to the cloud permits a departure from the old approach of designing once and maintaining for years. Instead, there is a shift to a frequent redesign concept of continuous improvement and ongoing cost-aware design optimization. There is sometimes a sense of a false trade-off between "lift and shift," where teams replicate the current on-premises architecture in the cloud, and a full "cloud native" modernization of the application (such as a move to microservices). In some cases, it may be worth the time and effort to fully refactor an application to meet specific business needs, including scalability, performance, resiliency, or other objectives. There is a full spectrum of V1 design options available to teams between "lift and shift" and full cloud native modernization. Some examples of design improvements short of full cloud-native architecture include:

- Upgrade End of Life (EOL) or near EOL versions of O/S, middleware, and application components that expose security risks
- Move to managed services such as load balancer, database, or caching service
- Improve resiliency with multi-AZ architecture and self-healing components
- Automate full-stack builds and deployments to support faster time to market
- Expanded test automation and coverage
- Add auto-scaling (for horizontal scaling or host swaps) and parallel processing for high-frequency workloads
- Use specialized instances for I/O, storage, or memory intensive workloads
- Support for blue / green deployments
- Add automated backup and archival functions, or automate disaster recovery / failover
- Supplement core operations tasks using AWS Lambda and AWS Step Functions
- Expand use of APIs
- Enhance data security and protection (such as Protected Health Information)

These should all be design considerations as you deploy on AWS. Critical changes should be implemented before moving to AWS. Less critical changes can be included in the optimization backlog to address after migration to AWS is complete.

The question is not "what is the best system architecture?" but rather "what is the best first step for AWS V1 design based on business objectives?" The time and effort to achieve these V1 improvements should be evaluated on alignment with business needs, available skills, and resources, and compared with other opportunities to invest in V1 improvements. Any backlog items can be included in future sprint objectives using a continuous improvement mindset.

Speed over perfection

The most critical aspect of transformation acceleration is to have an efficient decision-making process in place to set V1 design. Program acceleration objectives, cost / benefit trade-offs, and new capability delivery requirements will inform the degree of incremental change necessary for an appropriate V1 configuration.

It is essential to involve all relevant stakeholders and make the V1 AWS design decision quickly, then implement the agreed-upon plan and defer remaining optimization opportunities to the post-migration AWS backlog. The sooner application development teams are involved in designing, building, and operating on AWS, the more rapidly the promise and potential of operating on the AWS Cloud can be achieved. Think speed to market, experimentation, failing fast, delighting customers, reacting to market conditions, continuous improvement, and so on.

Business and application teams which can allocate 1-2 monthly release windows (depending on the complexity of the systems) in the short term to design, build, and deploy on AWS. They can experiment, innovate, and move faster after their business application portfolios are migrated to AWS. An investment in a “deep dive” on design with broad stakeholders will go a long way toward accelerating later build, test, and deploy steps in the migration process.

There are scenarios where there is a business case to complete application transformation prior to going live on AWS. Examples of this include assets that are beyond EOL and require an upgrade, refactoring of code to global pipeline, creating an API layer, or application code changes required to support multi-AZ deployment for tier 1 business-critical applications.

In any scenario where significant work is required beyond a “like for like” migration, this should be scheduled, budgeted, and planned, along with a cutover timeline, after the work is complete. If there are no resources available to complete the required work within the first two years of the program, teams should consider securing an exception if required, moving to AWS first, then implementing the required upgrades on AWS.

Is there value and capacity to transform? Do that. Are there competing priorities and limited capacity? Pick the simple path and move. Do not oscillate on what will be your first step to the cloud. Pick a path and take it. If it fails, implement your failback procedures, adjust, and try again. If you fail, you learn. You have made progress, rather than analyzing all your options and producing no incremental value.

Cutover to AWS is a “two-way door”. You have not made a hardware purchase decision that you’ll have to live with for years. This is very liberating, and unlocks team creativity when teams understand that they don’t have to know they got every decision just right.

The performance will tell you right away whether you have succeeded in testing, and again in application performance when you go live. You’ll learn more in production cutover. It’s very easy to adjust either on AWS (upsize on the fly) or fail back and try again. What teams should not do is belabor or second-guess V1 AWS design decisions.

Move thoughtfully but quickly through this process. Launch the build and test phases. Make adjustments if needed based on test cases, then deploy and tune, adjust, and optimize after deployment on AWS. In the rare case it does not work at all as expected, run your well-documented failback procedures, adjust your design, run code to deploy a new configuration, and try it again.

Pattern-based design approach

Based on customer engagements of the authors with AWS Professional Services, customers find that over 80% of their workloads can fit into a small number (~10-15) of re-useable architecture patterns to support application types. After these are designed and approved, let teams run with them. Teams can tailor and customize incrementally as needed, but subject matter experts (SMEs) and lead cloud architects need only to review incremental changes before build teams commence their work.

If build teams select a base template that does not require modification, they can begin build steps without further review. Review only necessary changes to the approved patterns, which is a light-weight review process, and empower teams to move fast with management by exception. Don’t build a large variety of patterns when it is not required. Keep it simple wherever you can. Save the effort on re-designing custom workloads (“big rocks”) that require significant change (such as mainframe application refactoring).

Launching a factory model and the first teams' capabilities

Develop a plan with clear milestone objectives

For large projects, leaders at Amazon are unwavering on the vision, but flexible on the details and the path to get there. Customers adopting cloud have an opportunity to set clear business targets of 100% migration of Windows and Linux-based platforms first, and to begin to sequence "first mover" teams to drive 50-100 applications at a time to AWS. They can then begin to fill in details as the factory velocity increases and more full skill-set migration teams are brought online. Ideally, business and technology teams should be organized around application portfolios of 50-100 applications, and can complete migration in less than one year. The application groups should be organized so that they can be managed by a small interdisciplinary team, and work with a dedicated set of application owners, testers and business partners / product owners.

Customers with portfolios of thousands of applications do not need a detailed plan for all the remaining business applications hosted on-premises to get started. The "boil the ocean" approach used by some companies and consultancies to do detailed planning for every last workload before getting underway in earnest for such a large portfolio slows progress. This is not an efficient use of resources, and by delaying the initial production migrations, removes opportunities for teams to "learn by doing".

Teams move more rapidly and can produce more sprint velocity on AWS only with experience, so the sooner teams begin moving, the sooner they can start to move up the learning curve. They need a framework, as described in this whitepaper, and a business unit to volunteer to join the transformation program journey. The rest of the details for what typically constitutes a multi-year program and thousands of applications can be determined in parallel to program launch. Not having all the answers is no reason to not get started. The best way to get through the transformation is to get started.

Set high-level objectives for each responsible team at the outset to provide a deadline to complete application migrations. Setting top-down aggressive goals, expressed in the number of applications in production on AWS on a quarterly basis, sets the line of sight while maintaining flexibility on the implementation. The detailed wave-by-wave planning can be filled in after teams are in flight and have a better grasp of the design decisions, and what transformation work will be required prior to cutover based on these decisions. While final wave planning can be deferred for 4-5 sprints (8-10 weeks) for a business unit work team, discovery and AWS design should be commenced as soon as it is practical after a migration team is organized around a set of business units or functional application suites.

Activate a "team-based" approach

Group the applications developed and managed by a single team, take 5-6 two-week sprints to activate the team and achieve migration velocity, and carry through until all the candidate workloads are migrated to AWS. The focus on moving small sets of applications rapidly from discovery to production on AWS limits the amount of "work in process".

For those workloads with a longer migration timeline or many components that need to be modernized with the move to cloud, and for suites that are tightly coupled, plan cutovers well in advance and with business alignment. There are several modes to coordinate design, build, deploy, and operate modes, and these can be managed by different owners if the consistent factory playbook is followed.

AWS does not recommend "cherry picking" easy applications across numerous teams. The context switching is not efficient. The effort to activate teams should be done once. "Cherry picking" takes longer for teams to get "all in" on AWS, where new capability development and operational efficiencies are

magnified in a single cloud environment. Applications selected for pilot should represent a set of similar applications in the portfolio, in terms of common requirements, complexity, and effort, and set the minimum standards for how subsequent migrations will be implemented.

Table 5 – Traditional vs. modern views of IT transformation planning and implementation

Traditional view	Modern view
Try to develop a detailed plan for the entire migration (all applications all waves) before starting. • Waterfall process (complete all design before staring migrations) • Sequential delivery (hand-offs) of infrastructure build, middleware configuration, code deployment, network configuration, and testing • Architecture teams operate separately from engineering, test, networking, and security	Activate a "team-based" approach. • Use migration to activate team capabilities • Reduce complexity by accelerating teams / portfolios through migration • Empowered teams with guardrails • Integrated two-pizza team design through cutover (Full stack product team or industrialized deployment for non-critical applications) • Sprint-based collaboration via rapid iteration through consistent dev, pre-production / QA, and production build and cutover

Deploy a "factory model" to accelerate your cloud-based business transformation

You can accelerate your journey to AWS by leveraging a "factory" model to distribute decision rights and increase velocity, while also maintaining the appropriate degree of configuration and architectural controls needed to drive re-usability, consistency, and enforcement of required global and local controls. The primary components of acceleration include:

- Agile delivery framework
- Structured "big rock" design sessions for complex applications
- Application infrastructure pattern development for re-usable architectures
- Templatized application documentation for design, build, and deployment phases
- Documented quality standards to increase build, automation, and deployment functions that can be distributed, reducing SME bottlenecks
- "Build faster" approach applied in specific cases, where it may be effective to split first-time application build documentation and testing in lower environments from incremental deployment automation in higher environments
- Clear operations integration plans and acceptance criteria for seamless transition to operations teams where applicable
- Observability strategy to measure performance, inform optimization priorities, and rapidly diagnose issues

The AWS migration factory approach is built on best practices from hundreds of AWS enterprise migrations, and can be specifically tuned to corporate standards. It is applicable to various patterns and paths, whenever a high degree of standardization can be propagated.

A key part to any transformational approach is "structure with flexibility". Structure includes minimum standards for completeness of design, exit, and acceptance criteria for each build stage, complete documentation including quality infrastructure as code instructions, and operational readiness verification. Regardless of team configuration, factory model, or deployment paths, the process of moving any application to AWS will incorporate the same standardized steps of discovery, design, build, deployment, integration, cutover, and operation.

How the build, deployment, and operations tasks will be shared or owned end-to-end will vary by team, but must meet all the criteria regardless of path. Another strategy to speed builds is to build and deploy consistently across development (Dev), QA / test, and production environments on the V1 path to migration. This may overbuild your lower environments on V1, but after cutover these lower environments can be re-sized or re-configured for cost efficiency.

For V1, speed is gained by consistency of builds. In the dev environment, you test and begin automation of build. In QA / test, teams will establish third-party or external connections if required, and run test suites in this deployment (mirroring production configuration). Production build, validate, and deploy teams may find any last-mile networking or other issues to solve. The cutover itself will usually be a non-event.

For each stage of deployment, there will be tested failback plans in place as well as backup, recovery, and disaster recovery (DR) procedures based on protection needs. Teams with clear acceptance criteria and runbook cutover timelines are empowered to make adjustments or rollback as needed. In every case, clear communication with business partners on cutover timeline and testing windows is the key to successful planning. AWS migration teams help customers develop and implement these plans, and continuously adapt and improve customers' migration factory practices tailored to specific business needs.

Scaling your cloud transformation program

Teams do not need to be configured or operate identically, even in a well-defined "factory model". Consistent with agile principles, teams should be empowered to find the best ways to collaborate within their teams, even when working in a large structured migration program.

Table 6 – Traditional vs. modern views of implementation controls

Traditional view	Modern view
• Use traditional siloed operating model • Command and control traditional wave planning • Freeze change / baseline and protect initial scope / planned activities • Maintain rigid separation of individual roles and responsibilities • Enforce "hand off" approach through quality gates	• Enable team autonomy with guardrails • Use migration to activate team capabilities • Reduce complexity by accelerating teams / portfolios through migration • Run what you build (where it makes sense) • Self-service access to reusable assets for speed and security • Automated controls and code scans built into foundations

Launch a cloud enablement engine to foster engineering best practices and reuse

To meet your cloud transformation objectives, take specific actions in a number of tracks:

- Launch detailed on-premises discovery to collect baseline dependency, sizing, and performance date for the entire environment
- Establish program, governance, and factory operating mode
- Assign core teams and launch sprint planning for the first one or two portfolio groups to move next, and begin planning for portfolio segment migration sequencing over the course of the full program
- Begin detailed design on complex applications ("big rocks") for initial portfolio move groups
- Set up an SME team to assess entire portfolio for systems with tight coupling to mainframe and midrange systems
- Begin development of a plan to decouple, resolve, or remove these dependencies

Launch a parallel cloud enablement engine (CEE) to:

- Evaluate business case options
- Review application roadmaps and help determine whether to refactor, retire, emulate, or decommission legacy mainframe and midrange platforms
- Translate DB2, SAS, Oracle, or Teradata legacy data workloads to modern cloud services

AWS customers have access to a wide variety of quick-starts, migration patterns, open-source vendor architecture patterns, and a wealth of other patterns available to move quickly. Efficient teams who look

for reusable patterns, perform careful security reviews, and build on top of prior work will move faster. Customers who set up a CEE with a pattern library and light review process can enable multiple teams across their enterprise to move quickly, shortcut non-differentiated design work, and focus on making improvements to their application performance.

AWS often works with customers to scan the application environment, identify common architectures, then develop complete designs for these patterns. This pattern-based approach can often cover a large portion of portfolio that might include simple vendor applications, application with low rates of change, or applications moving to the cloud that are on a sunset or ringfence (no planned investment) path. With this approach, builders pick up an approved security and networking-approved template including gold AMIs, make minor adjustments, get approval on any requested changes to pattern if required, then build and go. This unlocks velocity from your teams, as the sprint flow for these pattern-based deployments can focus on resolving hard-coding issues, connection issues, or other issues found as teams progress through each environment cutover.

With these templated accelerators and common design documents, design reviews can easily be managed by a small team of cloud engineering SMEs. It is feasible for a team of expert cloud engineers using this templated approach to review all design patterns and to suggest changes, revisions, optimizations, and pattern re-use opportunities. Simultaneously, these reviewers can ensure that corporate standards are met for design and resiliency, and aligned with application tier and business criticality. If this function is not entirely centralized in this program, distribution of these review tasks by application type, technology, or business unit would all be feasible as appropriate.

A CEE that supports learning and re-use, people change, cloud adoption, and innovation is an important enabler to accelerate enterprise transformation. Migrating to the cloud provides an opportunity to re-think the IT architecture function, and from a traditional central design team, to federate architecture work, drive secure pattern re-use, set standards, cross-pollinate teams, and help solve the most challenge business problems.

Teams that use AWS Service Catalog to design, review, and approve re-useable capabilities, or stacks, enable teams to move faster by presenting secure, re-usable assets and only reviewing incremental changes to approved patterns, if required for a particular use case. For more information, see Working with stacks.

Launch a people change strategy on day 1

In 2018, Gartner found that 88% of enterprise IT organizations have a cloud-first strategy, yet 86% of enterprise infrastructure spent is still directed to technology on-premises. While large organizations want to take advantage of cloud benefits, they have not undergone the necessary transformation within their organization to be successful. The existing culture, process, and tools block them from going “all-in” with AWS, even when they understand it is the right move for their business.

The lack of cloud skills and experience is a particularly difficult issue for enterprises to overcome. IT teams struggle to learn new skills while managing their current responsibilities. Many IT organizations are unable to recruit new cloud talent. This leaves leaders with a difficult decision: “Do we slow down delivery to give our teams time to transition to the cloud?”

The reality is that teams can be activated on AWS in a short period of time. The fastest way to achieve this is to learn by doing, by moving production workloads to AWS. With infrastructure as code, re-use of patterns, and under the guidance of AWS, AWS Partners, and internal expertise, it is possible to quickly enable teams on AWS. Teams engaged in migration efforts can sometimes get up to speed on AWS, infrastructure as code, and automation within 12-16 weeks. Supplementing on-the-job training with role-specific training paths and investing in educating business stakeholders helps teams succeed. See Training and Certification for more information on role-specific training paths.

You can do much more on AWS with fewer people compared to traditional siloed enterprise, on-premises operations. The key to transformation is to ignite your people’s curiosity, communicate the value proposition to learn new skills, and describe a new model where your staff can spend more time building and less time on non-differentiated heavy lifting.

People enablement through the migration process can also enable transformation in delivery and ownership to align with speed-to-market and better IT and business collaboration. The migration process is a good opportunity to re-evaluate the operations posture, including, in some cases, a move to a "run what you build" services model where the core development team may have "full stack" responsibility, including operations performance and improvements. There is no better place for experimentation with new roles and responsibilities than during the transformation program.

For a large organization, there may be a number of adaptations of teaming models, but several foundational items are common. As described earlier, such standards can be established through a central CEE or a dedicated project organization.

Team capabilities, working style, and technical obstacles will vary by team and application portfolio. AWS has learned from large-scale migrations that team members with an appetite to learn by doing on AWS can rapidly increase their confidence, skill set, and capabilities.

In addition to reinforcing the collaboration, agile methodologies, and move towards the DevOps model where appropriate, your transformation program democratizes technical skills across the IT lifecycle. An operations engineer can learn DevOps skills. Application developers can assume ownership for infrastructure provisioning and operation. Test engineers can help design a migration plan that improves test coverage and code scan automation using the corporate CI/CD toolchain. Everyone can adapt the mindset of "security is job zero". This process empowers teams, broadens opportunity, improves collaboration, and unlocks the creative potential of joint-disciplinary teams. The key is to get moving and gain cloud skills from this experience, fail, experiment, innovate, and delight customers.

Not everyone will be open to change. Not all staff will embrace the opportunity. Some staff may hold on to control of specific steps and processes with the goal of improving job security. Good leaders will identify and coach those who are blockers to change, use the process to cross-train and share tasks, and close documentation gaps to reduce the need to call an SME on weekends or vacation. This may also require adjustment of team members if some are not aligned with new ways of working.

There is an opportunity to move out of traditional "hero" mindset, where the most talented staff become indispensable, but are also a bottleneck and a single point of failure.

By improving documentation, automation, and using infrastructure as code to democratize build functions, a company's IT expertise can focus on building value and business capabilities, and spend less time on non-differentiated tasks and fire-fighting.

There is a value proposition for employees. Late-night and weekend coverage for operations can be delegated, and team members can take a vacation without concern. This should be considered a key aspect of a company's resiliency strategy, and must be supported by leadership to expand the definition of impact and value of their most talented employees.

Work closely with finance teams to shift to cloud

Budgeting a large cloud transformation program requires new skills. Transitioning from on-premises financial management of IT spend to AWS presents a great opportunity, using services tagging discipline to provide transparency of system costs, and architectural and resiliency cost / benefit trade-offs.

Organizations are initially uncomfortable with the shift from the precision of the cost of acquiring on-premises hardware for 4-5 years to the more flexible, adaptable, uncertain world of cloud cost management. With a disciplined migration approach, companies have achieved cost savings while migrating to AWS. They have enabled ongoing optimization as they right-size deployments and unlock additional value from moving faster and spending time on activities that create value. AWS recommends the *How to Think Like a Digital CFO* whitepaper for more information on the broader topic of cloud financial management.

Levers to improve the frequency of iterations and time to value

Without hard timelines or top-down aggressive goals, teams often lack a sense of urgency or clear deadline to complete their cloud transformation. This contributes to an extended period of "dual operations". As long as the longer product or portfolio teams are operating across two or more platforms, an extended period of increased complexity is created. There are many levers to increase the frequency of iterations and accelerate time to value. Some common AWS best practices are as follows:

- **If you can't decide on a migration path, pick the easy one** — If you can, migrate with minimal refactoring, then continue transforming on AWS. It is easier, quicker, and simpler to tune, experiment, or transform architecture on AWS. Evaluation of performance data will also inform what theoretical configuration changes may be effective, and changes can easily be scrapped if they are not performant or cost effective. Pace yourself and your teams, and adopt a crawl-walk-run mindset on design decisions and operational capabilities. Create an optimizations log for phases post cutover.

- **Consider the automation path** — Use the CloudEndure rehosting pattern or the VMware Cloud on AWS relocation pattern for non-differentiated workloads with low rate of change and low modernization value potential. Those workloads can still be re-factored once on AWS. This can create capacity on-premises, help avoid hardware refresh cycles on-premises, de-risk your migration, and focus limited skilled resources on re-deploying or re-factoring workloads with high value potential.
- **Address vendor applications early in the process** — There is widespread inclination to leave vendor applications, whether vendor-installed or customer-installed, until the end of the migration. A better approach is to schedule necessary vendor support and begin vendor-installed application migration at the start of the migration. In all but the most complex installs, vendor applications can be decoupled from other work and single-threaded after the initial build is well documented. Identification of dependencies early in the process is the key to avoiding a "pile-up" of work at the end of a team's migration journey. These dependencies include contractual due diligence, obtaining vendor commitment for the migration timeline, evaluating automation opportunities to reduce time to deploy, and dependencies on single resources.
- **Build consistently by building on initial builds for all environments** — Parallelize build and automation to "build faster". Keep the architecture consistent from development through non-production to production for the first deployment. After cutover on AWS, the environments can be optimized to reduce costs. Keeping consistency across builds removes a number of variables and greatly accelerates the path to production. Application cutovers in production is key for migration factories. Be sure to follow up by deprovisioning and/or right-sizing lower environments after successful cutover to ensure costs are optimized.

  When you need to "build faster" to stay within a specific time frame, parallelize paths for infrastructure build, application deployment, testing, and automation. After the design is completed, this process allows the infrastructure as code with AWS CloudFormation Templates (CFT) automation and the application installation, configuration, and testing to occur in parallel. Specifically, the first development build of the infrastructure can be deployed manually, and the application teams can begin test deployment in Dev immediately. While the application teams are developing the initial install pattern, the infrastructure code can be automated using a CFT deployed by the pipeline.

  With the consistent build deployed in non-production, the application teams can deploy incremental automation for configuration and code deployment. At the end of this stage, testing can be run on the full stack installation. After all use cases and UAT is validated and documentation is complete, the production build can replicate the non-production procedures, then final validation and cutover can be

implemented. This enables teams to move more rapidly and complete more work in parallel than with a purely sequential staging strategy.

- **Borrow a monthly cutover window from application teams for migration testing** — Planning far ahead for validation and release planning and getting business buy-in will solve a lot of problems. With enough advance notice, developers and testing engineers can plan accordingly and avoid conflicts with test resources schedules, which may be the biggest risk to schedule slip when test resource conflicts arise. Additionally, developer and test teams, with enough runway, can plan incremental automation of build, configuration, and testing steps to land in a more resilient posture on AWS, moving to infrastructure as code, adding auto-scaling groups, and reducing manual deployment procedures.

Conclusion

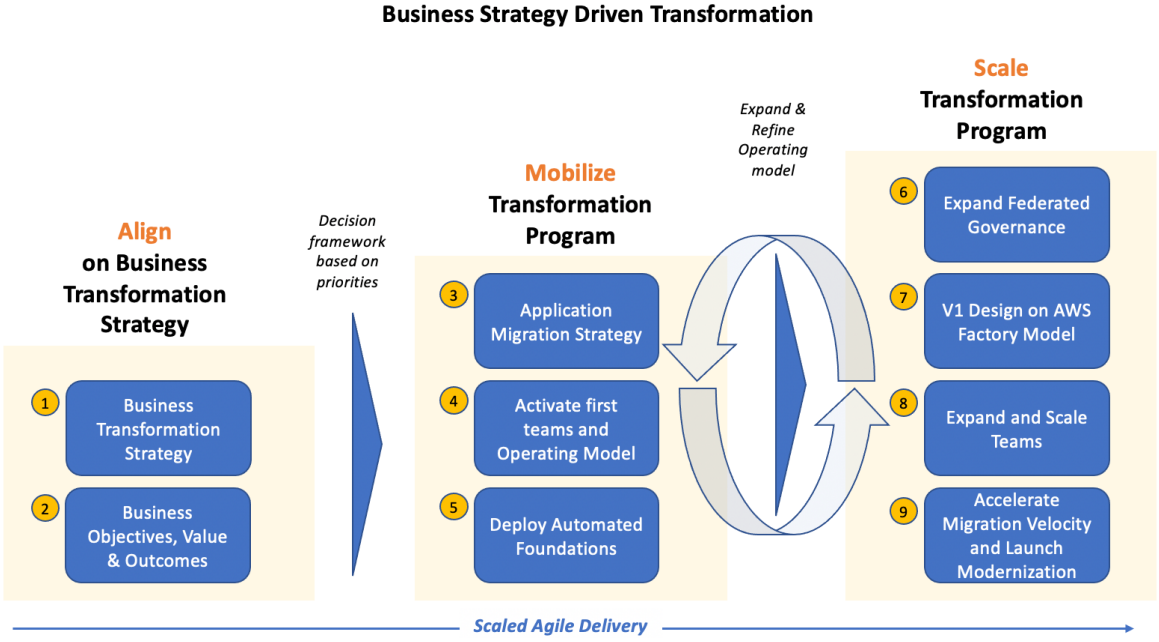
Beyond achieving the financial business case on IT spend, the two main questions successful companies ask during their cloud journey are “How do we move faster?” and “How do we enable our builders?”. Efficient decision-making allows you to accelerate the transition and improve speed to market for new capabilities. Successful companies enable builders through automation of deployment pipelines, secure integrated development environments, compliance and operations, and by having the right preventive and detective controls in place.

The advantage of the cloud at enterprise scale operations is that companies can replicate standards and controls at the platform account level, manage compliance with standards from a central or federated model, and then empower teams to move. With enterprise guidance for documentation, pattern approval, code scans, and other controls on the path to production through discovery, design, build, validate, deploy, and operate steps, teams can address the business problem at hand in a variety of ways. Valid strategies include building using cloud-native capabilities, refactoring applications, increasing automation and resiliency, moving towards services architecture, leveraging managed services, and containerization.

Without clear, aggressive goals and business ownership, teams languish in decision paralysis with options ranging from “do nothing” and leave on-premises, move as-is, or complete refactoring. Critical to taking the right first step, and moving with speed, is to have a clear V1 design process, and clear ownership of the decision-making rights with small teams of empowered stakeholders.

"We try to create teams that are no larger than can be fed by two pizzas," said Jeff Bezos. "We call that the two-pizza team rule."

Empower your teams to make these decisions, push them to move quickly, experiment, and fail. Enable collaboration and do the hard change work to distribute decision-making. Break up the traditional sequential “hand off” approach to delivery. With the north star vision of high performing teams moving towards “full stack” responsibilities (run what you build), and cost and performance driven architecture, there is no limit to what you can deliver to your customers.



Business-strategy-driven transformation

Document history and contributors

To be notified about updates to this whitepaper, subscribe to the RSS feed.

update-history-change	update-history-description	update-history-date
Initial publication (p. 23)	Whitepaper first published	May 11, 2021

Contributors

Contributors to this document include:

- Andrew Haggard, Principal Enterprise Service Manager, AWS Professional Services
- Rostislav Markov, Principal Customer Delivery Architect, AWS Professional Services
- Nirav Kothari, Principal Customer Delivery Architect Leader, AWS Professional Services

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.