AWS Cloud Adoption Framework

Process Perspective

February 2016

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 AWS 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 paper covers the Process Perspective. The considerations in this perspective can help you ensure that all your enterprise processes are in place to plan, implement, and operate cloud-based IT capabilities.

Introduction

The Process Perspective covers activities across the IT lifecycle for cloud adoption. The focus is on managing IT initiatives as a portfolio to optimize investments, deliver services that meet quality objectives, and carry out work through well-defined programs and projects. For cloud-based software development, you can use agile and iterative lifecycles to deliver functionality incrementally and to catch and fix defects early. You can use continuous integration/continuous delivery (CI/CD) practices to automate building, testing, and deploying software. You can also use CI/CD to automate operational processes to improve the resilience of the solutions and reduce manual effort. Figure 2, gives you a high-level look at the process perspective components, activities, and artifacts that are discussed in this paper.

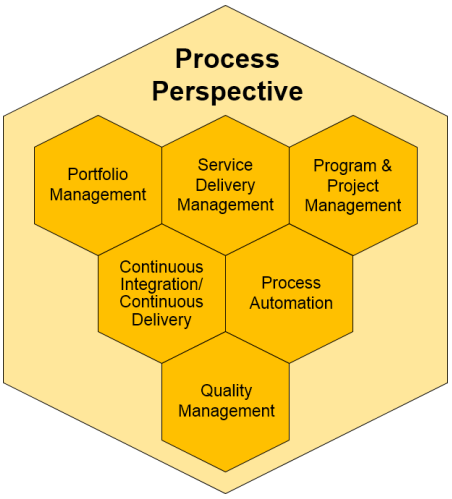

Figure 1 Components of the Process Perspective

AWS CAF Process Component	Activities	Artifacts
Portfolio Management	▪ Define economic framework for decision making ▪ Shorten planning cycles ▪ Apply goal-oriented requirements engineering techniques to get to target conditions	▪ Enterprise IT Domain Diagram ▪ Enterprise IT Capabilities to Enterprise Domain Matrix ▪ Economic framework that includes the cost of delay ▪ Impact Map
Service Delivery Management	▪ Adjust existing service delivery processes ▪ Institute Lean-Agile engineering practices ▪ Target a DevOps culture	▪ Lean-Agile engineering techniques ▪ DevOps practices
Program & Project Management	▪ Establish a portfolio to manage all business and IT capabilities ▪ Integrate cloud-based IT services into your current Sourcing Model	▪ Enterprise Portfolio spreadsheet ▪ Sourcing Model ▪ AWS Application Migration Method ▪ AWS Migration Factory
Continuous Integration & Continuous Delivery	▪ Document current SDLC practices ▪ Develop approach for teams to adopt CI/CD practices via continuous improvement practices ▪ Implement Delivery Pipelines	▪ CI/CD Tools Matrix
Process Automation	▪ Document current processes using Value Stream Mapping or other techniques ▪ Identify and prioritize process optimization opportunities ▪ Identify and prioritize process automation opportunities	▪ Value Stream Map ▪ Activity Based Accounting
Quality Management	▪ Institute continuous improvement practices ▪ Incorporate continuous improvement into all your IT practices	▪ Value Stream Mapping ▪ Theory of Constraints ▪ Root-Cause Analysis ▪ Improvement Kata ▪ Coaching Kata ▪ Plan, Do, Check, Act (PDCA)

Figure 2: Process Perspective Activities and Artifacts

Service Delivery Management

The Service Delivery Management component of the AWS CAF Process perspective promotes leveraging people, processes, and technologies in ways that help you optimize how you deliver business outcomes. Consider using the following approach to start taking advantage of cloud-based IT services to optimize your service delivery processes:

1. Document your current service delivery processes.
2. Review the target outcomes you defined during your cloud strategy development activities.
3. Identify the minimum process modifications required to support the outcomes defined in your cloud strategy.
4. After you understand your desired target state, start working through internal change management processes to begin the change process.

Figure 3 provides an example of an IT lifecycle and the key areas for review to determine your target state.

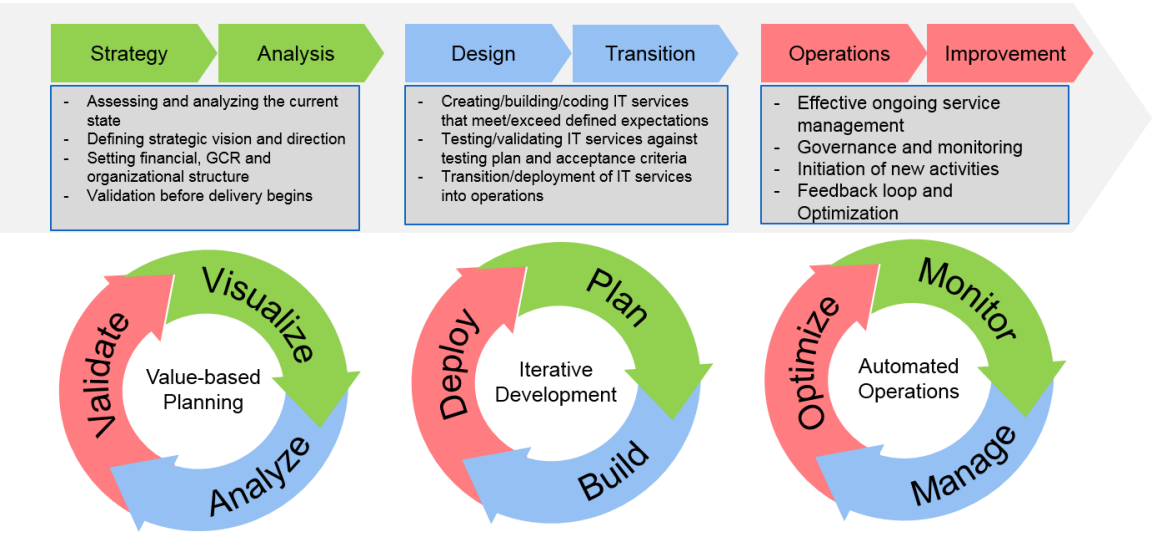

Figure 3: IT Lifecycle

This approach will ensure you have done the minimum required to adopt successfully cloud-based IT capabilities into your service delivery processes.

Transform Your Business with DevOps

To achieve the greatest value from adopting cloud-based IT services, you should consider adopting Lean agile engineering practices and a DevOps culture. As

Damon Edwards has stated, "The whole point of DevOps is to *enable your business* to react to market forces as quickly, efficiently, and reliably as possible."[2] It does this by focusing on optimizing your service value streams that encompass every step from idea inception through customer consumption. This type of transformation has enabled Amazon and many other companies to achieve innovation at ever-increasing scales.

You Build It—You Run It

Leveraging DevOps in a cloud-based IT delivery model provides many benefits. Werner Vogels, the CTO at AWS has said, "You build it, you run it." In a 2006 interview, Vogels describes the value of moving away from an operating model where separate teams develop and operate services to a model where a single team both develops and operates a service. "The small-team concept means that you have a continuous feedback loop where you try to understand the impact for the customer."[3]

Consider making the following improvements to realize the value DevOps can bring to your IT operating model:[4]

- **Design for production**—"Run what you build" forces development teams to think about how their software is going to run in production as they design it. This can help your teams avoid the last minute scrambling that often occurs when teams try to force-fit what they've built to a production environment to meet a deadline. This is an all too common occurrence that materially hurts quality. You change something at deployment time to address something that's different between production and development. Next, you run what you think are the relevant tests, and later discover that this change caused a bug somewhere else in the system.
- **Encourage greater employee autonomy**—The "you build it, you run it" mentality encourages ownership and accountability and can lead to more independent, responsible employees and greater potential career growth in the organization.
- **Implement greater transparency**—Your teams will naturally want greater transparency in the environment, so it would be helpful to implement proactive monitoring so they can identify issues and concerning patterns before they become widespread problems. Transparency should make it much easier to find root causes for issues that still make it through, ultimately leading to increased uptime and improved service quality.

- **Build in more automation**—Developers hate repeating manual tasks. So, instead of having developers repeat tasks in production to address an issue, automate things along the way so that developers can discover root causes of issues earlier.
- **Continually improve operational quality**—Continual improvement practices, feedback loops, and test-driven development, along with automation and transparency can lead to higher quality services for your internal and external customers.
- **Learn more about satisfying customers**—"Run what you build" forces the entire IT team to understand more about the customer. Knowledge will no longer be limited to a product or sales team. Developers can include customer feedback metrics in their software to quantify the value of each feature to their customers.

Why Culture Matters

The Puppet Labs 2015 State of DevOps Report acknowledges, "Culture is the most important ingredient of DevOps."[5] The report measured a company's culture and IT performance and demonstrated a strong link between the two. Good DevOps practices generate high trust in the organizational culture. These practices are a factor in what Ron Westrum calls a "generative performance-oriented" culture. Figure 4 shows Westrum's comparison of three different organizational cultures and their characteristics.[6][7]

Pathological Power-oriented	Bureaucratic Rule-oriented	Generative Performance-oriented
Low cooperation	Modest cooperation	High cooperation
Messengers shot	Messengers neglected	Messengers trained
Responsibility shirked	Narrow responsibilities	Risks are shared
Bridging discouraged	Bridging tolerated	Bridging encouraged
Failure leads to scapegoating	Failure leads to justice	Failure leads to inquiry
Novelty crushed	Novelty leads to problems	Novelty implemented

Figure 4: Typology of Organizational Culture (Westrum, 2004)

Figure 5 provides additional practices from the Puppet Labs report that will help you achieve a strong DevOps culture.

Characteristics of a Generative Culture	DevOps Practices
High Cooperation	**Cross-functional teams -** Many organizations create cross-functional teams that include representatives from each functional area of the software delivery process (business analysts, developers, quality engineers, ops, security, etc.). This allows everyone to share the responsibility for building, deploying, and maintaining a product.
Messengers trained	**Blameless post-mortems -** By removing blame, you remove fear; by removing fear, you enable teams to more effectively surface problems and solve them. Mistakes happen. Holding blameless postmortems is a valuable way to learn from mistakes.
Risks are shared	**Shared responsibilities -** Quality, availability, reliability, and security are *everyone's* job. One way to improve the quality of your services is to ensure that devs share responsibility for maintaining their code in production. The improvement in collaboration that comes from sharing responsibility inherently reduces risk: With more eyes on the software delivery process, it is a given that some errors in process or planning will be avoided. Automation also reduces risk, and choosing the right tool can enable collaboration.
Bridging encouraged	**Breaking down silos -** In addition to creating cross-functional teams, consider co-locating ops with the dev team and including ops in planning throughout the software and delivery lifecycle.
Failure leads to inquiry	**Blameless post-mortems -** Response to failure shapes the culture of an organization. The more you focus on the conditions in which failures happen, as opposed to blaming individuals for failures, the closer you'll get to creating a generative culture.
Novelty implemented	**Experimentation time -** Giving employees the freedom to explore new ideas can lead to great outcomes. Some companies give engineers time each week for experimentation. Others host internal hack days or mini-conferences to share ideas and collaborate. This is how many new features and products have originated, and it shows how much value employees can generate for an organization when they are released from habitual pathways and repetitive tasks.

Figure 5: How to create a generative culture

Considerations

- **Do** consider adopting lean agile engineering practices and a DevOps culture to transform your business.
- **Do** analyze your existing service delivery processes to determine which must change to support adoption of cloud-based IT services.
- **Do** define key metrics that the CIO will report on to the board.
- **Do not** implement and maintain separate processes and tools for cloud-based versus on-premises based IT services.
- **Do not** set service-level agreements (SLAs) unless you have activities in place to test for compliance. Game day testing is one technique that allows you to test SLA compliance by using automated scripts to introduce failure scenarios into your systems.

Portfolio Management

The Portfolio Management component of the AWS CAF Process Perspective provides executives with a method for managing new investments and identifying how well existing investments are delivering on desired outcomes. If you are doing this today, then you just need to adjust your portfolio practices to include cloud-based IT services. If you are not already doing this, then you can get started by creating an inventory of existing assets and prioritizing new products and services in a management platform like a service catalog.

Increase Frequency of Planning Cycles

Traditional portfolio management typically runs on a yearly rhythm with quarterly review and reprioritization. Consider moving to lean portfolio management practices that incorporate a continuous planning and management strategy to reduce the time to value in the delivery of solutions. Lean portfolio management uses a mindset similar to high-velocity stock brokering by having many projects that are very frequently aligned, re-prioritized, and re-budgeted based on their individual performance, value add, criticality, and dependency. For this approach to be effective, the practice of protecting "untouchable" programs and projects must be abandoned. By adopting this approach, you will gain the agility to change as business needs change.

A Service Catalog is a foundational enabler of the IT planning and delivery cycles. It contains details about the IT systems that support business capabilities and how those systems are sources and hosted. This information provides critical input for identifying and sequencing portfolio investments. Combining a robust Service Catalog with governance that ensures funding and flexibility to development teams can help to mature your portfolio management practices.

Using model-driven and data-driven approaches to testing specific re-prioritization scenarios will give you greater ability to shift portfolio prioritization at the same speed that the business changes. For example, using modeling techniques to test how pivoting on the cost of delay as the main prioritization key and impact to portfolio planning might yield valuable insight.

Drive Alignment Through Portfolio Governance

As part of an AWS cloud adoption strategy, you have an opportunity to tighten the partnership between business and IT planning and management efforts. By

creating a partnership between business and technology teams, you give development teams the opportunity to collaborate directly with business sponsors. This collaboration will lead to higher quality feature delivery and improved customer satisfaction.

Portfolio management should become your key governance body that controls the boundaries for why, what, and how demand traverses your IT lifecycle. The portfolio management, program management, and business operations teams should collaborate to maintain close alignment with the business. Figure 6 is an example of a governance model used to describe how various groups will collaborate to accelerate delivery of capability into production.

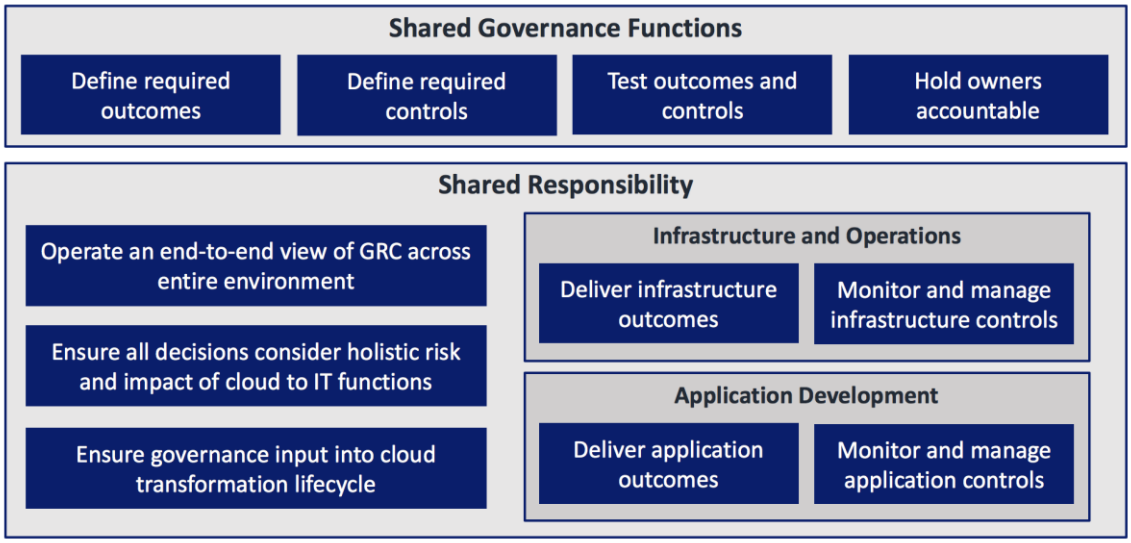

Figure 6: Governance Model

Ideally, shifting the portfolio management function to a center of excellence (CoE) function can provide teams with more autonomy and improve their responsiveness to changes in business requirements.

Think Sourcing Before Hosting

Many customers view AWS services as a set of technology tools that can enhance their existing toolkit. They learn about AWS compute, storage, and private networking services and then go about mapping their current application's components to these foundational services. This approach is valid, but it reinforces a builder-centric approach to delivering IT capabilities. We have found that customers who first consider how they source their business and IT capabilities achieve far greater business value using AWS.

The Capability Sourcing Model in Figure 7 guides you through a set of decisions to be made on the sourcing and hosting of your business and IT capabilities. We recommend that you either extend your existing sourcing model or customize this base model to meet your specific needs.

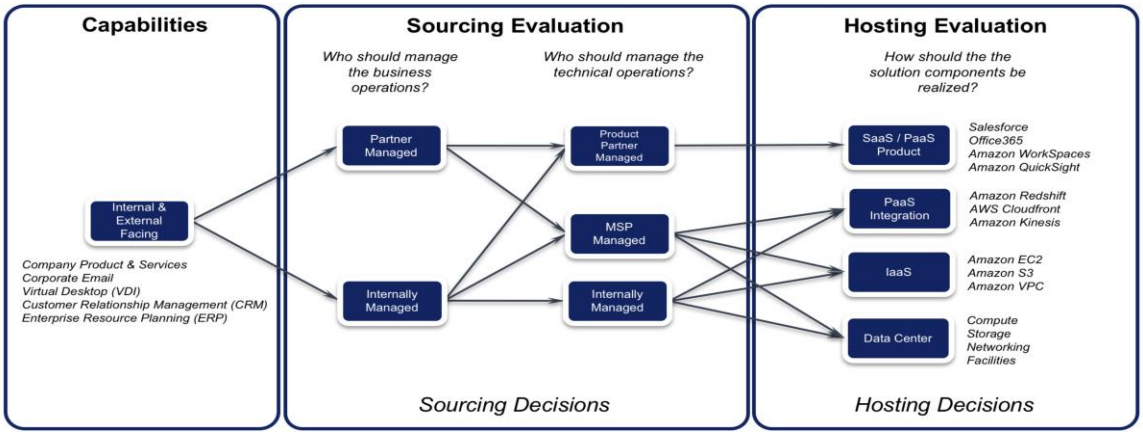

Figure 7: Capability Sourcing Model

We also recommend adopting a cloud first approach to sourcing. You should target Software as a Service (SaaS) options first. If SaaS is not feasible then consider Platform as a Service (PaaS), Infrastructure as a Service (IaaS), commercial off-the-shelf (COTS), or bespoken solutions, in that order. This approach will limit the amount of code that you must write and maintain over time and reduce the operational cost required to achieve your desired outcomes.

Gain Visibility and Collaboration

With the speed of change in business needs and changes made to the IT environment, management tools might be required to keep up with a more distributed portfolio management approach. This might require tooling to automate monitoring and management of changes to the portfolio.

Considerations

- **Do** minimize your code base while expanding features and functionality by keeping the sourcing strategy separate from the hosting strategy.
- **Do** set a principle of Software as a Service (SaaS) first, followed by Platform as a Service (PaaS), then Infrastructure as a Service (IaaS) to minimize the amount of code required to support all capabilities.
- **Do** incorporate a decision-making process that governs portfolio management.
- **Do not** use a monolithic and detailed planning process for portfolio management. Consider providing development teams a funding pool and using a higher-level oversight technique for portfolio management.
- **Do not** separate IT portfolio management from business portfolio planning and strategy meetings. Consider shifting IT portfolio management to a CoE resource shared by the IT organization and the business units.
- **Do not** make the portfolio management team responsible for the entire portfolio management process. Consider making the portfolio management team responsible for governance, management, and reporting on spend and the development teams responsible for planning execution towards stated outcomes.

Program and Project Management

The Program and Project Management component of the AWS CAF Process perspective promotes the provision of the right balance of autonomy and governance so you can increase the value you gain from adopting cloud-based IT services. A lean approach to project portfolio management gives program teams the autonomy to make decisions, enabling them to adjust quickly to changing business needs. They have the freedom to scope the size of projects, within governance boundaries, and can choose the methodology and tools they use to deliver it. This automony makes them more accountable for delivering solutions that best meet business objectives.

Outcome-Driven Planning

You should develop new or evolve existing capabilities by focusing on the outcomes that you want to achieve. You can use goal-oriented requirements engineering techniques to guide you. Your understanding of what the product should be will evolve as you learn from the results of each release, so spend as little time as possible determining what your minimally viable product should be. This approach will help you reduce upfront funding and planning.

Incorporating customer feedback loops into your product will allow you to make decisions on how the product should evolve, based on validated learning. You can use A/B testing to run controlled experiments and test your hypotheses to determine which options best meet your customer's desired outcomes.

Complex Enterprise System Challenges

We hear about many customer challenges associated with evolving their large enterprise and/or legacy systems (e.g., ERP, CRM, and others). A common approach to a major system rewrite is to create a multi-year program to develop the "new" system. You might have lived through one or more of these initiatives in your career and understand the time, cost, and risk associated with them. One effective approach to solving these challenges is to implement a "strangler application" instead of performing a "big bang" cutover. Martin Fowlers says that to use the strangler application approach you need "to gradually create a new system around the edges of the old, letting it grow slowly over several years until the old system is strangled".[8] Using this approach, you can evolve your biggest legacy systems using small, manageable projects.

Maintain Small Teams to Scale

Amazon uses small "two-pizza" teams to develop and operate our services and new features. These teams are made up of individuals that are experts in one or more disciplines; feature development, automation, testing, security, operations, product development, etc. Each team works independently to collect customer feedback, maintain a roadmap, deliver new functionality, ensure reliability, and meet the compliance and regulatory requirements for their service. Avoiding the communication challenges associated with large teams is a key factor in Amazon's ability to innovate at scale.

Considerations

- **Do** consider using small-batch project delivery techniques for new feature development. Use "rolling wave" program and project planning approaches, so planning is much more continuous.
- **Do** consider using project outcomes to develop portions of existing solutions rather that attempting to replace an entire solution.
- **Do** define suitable metrics to capture and communicate the effectiveness of lifecycle processes and activities.
- **Do** set ensure there is oversight to identify and resolve potential conflicts between service activities.
- **Do not** define SLAs without implementing compliance testing. Consider using game day testing techniques that introduce failures in systems to validate SLA compliance.
- **Do not** try to replace large-scale systems and solutions as a single project. Consider breaking large initiatives into multiple autonomous smaller projects with shorter timelines.
- **Do not** assume that you must maintain all existing processes. As you transform to cloud-based environments, consider changing processes.
- **Do not** use top-down approaches for all planning. Consider approaches that give development teams the freedom to plan how they will meet the desired outcomes agreed to with the business team.

Continuous Integration and Continuous Delivery (CI/CD)

The CI/CD component of the AWS CAF Process persective helps organizations focus on adapting agile software development practices for incremental delivery of functionality. The traditional waterfall model is a sequential software development lifecycle in which progress flows steadily downwards (like a waterfall) through the phases of Initiation, Requirements, Analysis, Design, Construction, Testing, Implementation and Maintenance. Modern agile software development involves incrementally developing working software through iterative planning, design, development, and testing.

Using the agile model, you can leverage CI/CD practices and tools to automate your software delivery lifecycle through automated builds, deployment, and testing. This leads to quicker delivery and continuous improvement of your products.

Achieve Continuous Integration

To achieve CI the goal is to create working software with every change committed to your project's version control repository. Consider using trunk-based development to reach this goal. With trunk-based development, everyone commits changes to the trunk daily, and each check-in triggers a build and automated tests. When a test uncovers a regression, the developer or team responsible for that feature must stop current feature development and resolve the regression. If they cannot resolve the regression within a few minutes, then they must revert the change.[9] If all your development teams follow this approach, then they have achieved CI.

Implement Deployment Pipelines

Some experts define the focus of Continuous Delivery (CD) as "how all the moving parts fit together: configuration management, automated testing, continuous integration and deployment, data management, environment management, and release management."[10] Provisioning in this manner will allow the environment to support your teams' ability to deliver as they gain experience working in CI/CD environments. Paul Duvall[11] identifies three purposes for the deployment pipeline:

- Visibility: All aspects of the delivery system - building, deploying, testing, and releasing – are visible to all team members promoting collaboration.
- Feedback: Team members learn of problems as soon as they occur so they can fix issues as soon as possible.
- Continually Deploy: Through a fully automated process, you can deploy and release any version of the software to any environment.

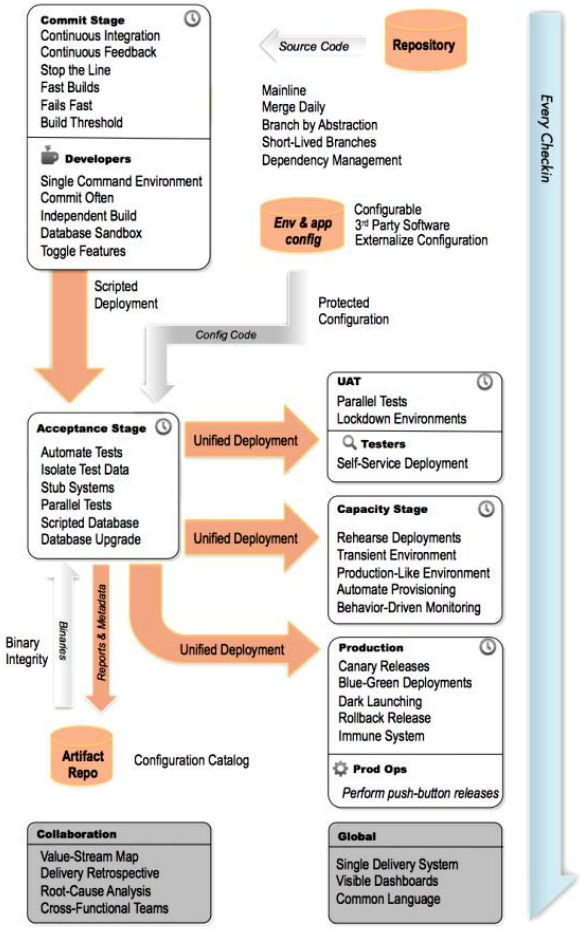

Figure 8: Components of a Delivery Pipeline (Duvall) [8]

Deployment Is Not Release

It's important to distinguish between a technical decision—to deploy a product, and a business decision—to release a product. Jez Humble notes that a release is "the process of making a feature or set of features, available to customers." A software release is a business decision. He defines a deployment as "the installation of a given version of a piece of software to a given environment". A deployment to production is a technical decision.[12]

System Level Deployment

Consider using blue-green deployment, which is a continuous delivery technique that reduces the cutover-time of releasing to production. With this approach, you maintain two production environments that are as close to identical as possible. At any point in time, one of these production environments, blue for example, will be active. As you prepare the next release, you will use the other environment, green in this case, for final testing. After testing is complete, you will redirect production traffic from blue to green making it active. Blue then becomes idle, until the next release is ready for final testing.

Feature Level Deployment

Dark launching is a continuous delivery technique that you can use to release one to many features or components. Developers use feature flags to control which customers have access to the features. This technique enables A/B testing and the slow ramp of features to customers.

Considerations

- **Do** consider transitioning to a Dev/Sec/Ops type development environment that includes development, security, and operations skills on the development team.
- **Do** take advantage of direct customer data by using A/B testing techniques to validate hypotheses.
- **Do** create a robust process that can manage thousands of daily changes.
- **Do not** set up your AWS environment in a manner not designed for CI/CD. Even if you shift to a CI/CD environment over time, set up your AWS environment to support CI/CD.

Process Automation

The Process Automation component of the AWS CAF Process perspective takes the value an organization gets from adopting the cloud to the next level. The cloud removes much of the undifferentiated heavy lifting associated with managing traditional IT. Automation offers an organization additional benefits that extend into many different IT disciplines.

There are a number of mature tools that you can use to automate your processes—AWS CloudFormation, Chef, Puppet, Ansible, and others.

You should consider process automation if you have questions such as:

- "How can I save money with the cloud?"
- "How do I make my applications secure?"
- "How do I avoid being locked into one platform?"
- "How can I move faster?"

Stephen Orban has discussed the benefits of automation in detail. His findings include the following:[13]

Efficiency—As an executive, one of your primary responsibilities is to focus an increasing percentage of your resources on initiatives that drive revenue into the business. This is a main motivation for using cloud services—by reducing the amount of time you spend managing infrastructure you can repurpose those resources toward product development. By automating repeated tasks such as performing operating system patches, application deployments, firewall changes, and monitoring/alerting configurations, you will be able to devote more cycles to the development of features.

Elasticity—This is one of the most beneficial features of a cloud platform. Automated systems that you can reproduce using scripts can leverage AWS services like Auto Scaling to fluctuate capacity with demand. Capacity planning is generally very hard to do, and getting it wrong can be a large waste of capital and/or very frustrating to your customers.

Portability—When you can reliably reproduce a system on one architecture or infrastructure it will be much easier to reproduce it on another. If you need to

change where you host a system, it will be far less effort to modify the automation scripts than if you had to customize each platform manually.

Security—Automating good security practices into your systems makes it much more likely that your system will be uniform as it scales, removes the element of human error creating an exposure, and allows you to roll out security improvements (like any feature) quickly and with confidence. Some of the better-architected systems we see leverage automation in phases. During each phase a specific action is performed—updating an operating system, downloading code from an internal repository, bringing the application up on the Internet, while only allowing the connectivity needed for each phase. This helps make sure traffic coming in from the Internet doesn't have connectivity into corporate systems.

Auditability—Systems built using automation scripts tend to behave more predictably and have more informative logging. The sequence of actions taken for anything automated is documented by the code that performs those actions. This makes it much easier to describe to others—your team, business stakeholders, auditors, and regulators—who performed what action and when in your systems.

Recoverability—When it comes to recovery, there are at least two factors that work in your favor when building automated systems. First, as you gain experience with automating common actions, it will become clear how to implement automated health checks and common QA actions. Automation can then be used to roll back systems in flight that don't pass these checks. Second, as you reduce the friction to changing an environment to zero, you can become more aggressive with how you push out changes. It becomes less costly and time consuming to roll something out and either roll it back or roll out a patch on top of it than it does spending days running manual QA or debating the risks.

Adopting AWS services enables you to manage infrastructure changes in the same way that you manage software changes. Create the definitions for the infrastructure environments you need, and then maintain and store them as you would software code.

Considerations

- **Do** update and store infrastructure definitions in your code repository.
- **Do** create an automated service catalog.

- **Do** create scale-up and scale-down strategies prior to starting a cloud adoption initiative.
- **Do** create a multi-tenant provisioning strategy prior to starting a cloud adoption initiative.
- **Do not** wait for failure and react. Consider creating a disaster recovery strategy prior to starting a cloud adoption initiative.

Quality Management

The Quality Management component of the AWS CAF encourages the use of lean and agile principles and practices to help your organization deliver higher quality products, improve customer satisfaction, and achieve a more sustainable competitive advantage. Following these principles will help you engage all your employees in continual improvement and integrate the quality discipline into the culture and activities of your organization. Adopting cloud-based services to deliver your software products will be a key enabler of these outcomes.

Lean Principles

Toyota developed lean manufacturing just after WWII, and used it to grow into the world's largest carmaker, earning top marks for quality and innovation. Over the past two decades, there has been increased adoption of lean principals by businesses outside of manufacturing, to address similar business challenges. The five overriding lean principles[14] are:

1. **Identify Customers and Specify Value**—The starting point is to recognize that only a small fraction of the total time and effort in any organization actually adds value for the end customer. By clearly defining value for a specific product or service from the end customer's perspective, all the non-value activities—or waste—can be targeted for removal.
2. **Identify and Map the Value Stream**—The value stream is the entire set of activities across all parts of the organization involved in jointly delivering the product or service. This represents the end-to-end process that delivers the value to the customer. After you understand what your customer wants the next step is to identify how you are delivering (or not) that to them.
3. **Create Flow by Eliminating Waste**—When you first map the value stream it is typical to find that only about 5 percent of activities add value; this can rise to about 45 percent in a service environment. Eliminating this waste ensures that your product or service "flows" to the customer without any interruption, detour, or waiting.
4. **Respond to Customer Pull**—Understand the customer demand on your service and then create your process to respond to this demand. The goal is to produce only what the customer wants when the customer wants it.

5. **Pursue Perfection**—Creating flow and pull starts with radically reorganizing individual process steps, but the gains become truly significant as all the steps link together. As the process evolves more and more layers of waste become visible and the process continues towards the theoretical end point of perfection, where every asset and every action adds value for the end customer.

By following these five lean principles, you will implement a philosophy that will become "just the way things are done." They will help you ensure that you are driving towards the overall organizational strategy by constant review of your processes to ensure that they are constantly and consistently delivering value to your customer. This will allow your organization to maintain a high level of service while being able to grow and respond to changing business needs, and it does this through implementing sustainable change.

Agile

Agile refers to a set of values and principles defined in the Agile Manifesto[15], created in 2001. The Manifesto was a reaction to heavier weight software development methods, like waterfall, that the manifesto authors felt constrained their ability to deliver software that customer truly needed.

The values espoused in the Agile Manifesto include **individuals and interactions** over processes and tools, **working software** over comprehensive documentation, **customer collaboration** over contract negotiation, and **responding to change** over following a plan.

Agile has the following principles:

- Highest priority is customer satisfaction
- Welcome changing requirements
- Frequent delivery of software
- Business people and developers cooperating daily
- Build projects around motivated people
- Face-to-face conversation is best
- Progress measured by working software
- Sustainable development pace
- Continuous attention to technical excellence
- Simplicity
- Self-organizing teams
- Regular reflection and adaptation

One of the signers of the manifesto, Jeff Sutherland, states that Agile development is not a method in itself, but rather an umbrella term that describes several agile methodologies. Examples of these agile methods include the following[16]:

- Scrum or Kanban for management
- Extreme programing
- Continuous integration
- Continuous delivery
- Feature-driven development
- Test-driven development

Each individual agile methodology approaches these values in a slightly different way, but all of these methodologies have specific processes and practices that foster one or more of these values. Software teams that adopt these values and principles, and apply agile development methods are much more likely to achieve true agility.

Considerations

- **Do** adopt lean principles and learning kata to increase productivity and reduce waste across your organization, and improve customer satisfaction.
- **Do** adopt the agile values, principles, and methods to deliver software that customers want, faster, and with higher quality.
- **Do** use value-stream mapping to enable sustainable system improvement.
- **Do not** falsely assume that by adopting select agile practices you can achieve the outcomes that true agile adoption enables.

Conclusion

Acknowledging that your Enterprise business processes must change is a critical first step in understanding the effort required to achieve value from adopting cloud-based IT services. The next step is choosing your approach. You can choose to change just enough to ensure the sustainable operations of cloud-based systems. This will add value focused on IT operations. At the other extreme, you can choose a much broader transformation approach that can enable your business to thrive. The AWS CAF Process Perspective can help you choose the approach that is best for you.

CAF Taxonomy and Terms

The Cloud Adoption Framework (CAF) is the framework AWS created 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, the Process Perspective provides guidance on to help you ensure that all of your enterprise processes are in place to plan, implement, and operate cloud-based IT capabilities.

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

For example, *Service Delivery Management* is one component of the Process Perspective and adopting lean-agile engineering practices 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.

Notes

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

[2] Edwards, Damon. 2010. "DevOps Is Not a Technology Problem. DevOps Is a Business Problem. - dev2ops - dev2ops." Retrieved September 7, 2015 (http://dev2ops.org/2010/11/devops-is-not-a-technology-problem-devops-is-a-business-problem/)

[3] A Conversation with Werner Vogels - ACM Queue." Retrieved September 21, 2015 (http://queue.acm.org/detail.cfm?id=1142065).

[4] Orban, Stephen. 2015a. "Enterprise DevOps: Why You Should Run What You Build — AWS Enterprise Collection — Medium." Retrieved September 9, 2015 (https://medium.com/aws-enterprise-collection/part-3-in-the-enterprise-devops-series-why-you-should-run-what-you-build-c62f0990f4c3)

[5] abs, Puppet and IT Revolution Press. 2015. "2015 DevOps Report." (877). Retrieved (https://puppetlabs.com/sites/default/files/2015-state-of-devops-report.pdf).

[6] Westrum, R. 2004. "A typology of organisational culture. " Retrieved November 10, 2015 (http://qualitysafety.bmj.com/content/13/suppl_2/ii22.full.pdf+html).

[7] Humble, Jez, Joanne Molesky, and Barry O'Reilly. 2014. "Lean Enterprise: How High Performance Organizations Innovate at Scale." Retrieved May 17, 2015 (http://www.amazon.com/gp/product/1449368425).

[8] Fowler, Martin. 2004. "StranglerApplication." Retrieved May 17, 2015 (http://www.martinfowler.com/bliki/StranglerApplication.html).

[9] Duvall, Paul. 2011. "Continuous Integration - DZone - Refcardz." Retrieved September 6, 2015 (https://dzone.com/refcardz/continuous-integration).

[10] Jez Humble and David Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation", Addison Wesley Professional, 2010

[11] Duvall, Paul. 2011. "Continuous Delivery - DZone - Refcardz." Retrieved September 6, 2015 (https://dzone.com/refcardz/continuous-delivery-patterns).

[12] Jez Humble and David Farley, "Continuous Delivery: Reliable Software Releases through Build, Test, and Deployment Automation", Addison Wesley Professional, 2010

[13] Orban, Stephen. 2015b. "Make Automation a Key Part of Your Cloud Strategy — AWS Enterprise Collection — Medium." Retrieved September 9, 2015 (https://medium.com/aws-enterprise-collection/make-automation-a-key-part-of-your-cloud-strategy-8eea07b0986c).

[14] Lean University. "The Five Principles of Lean Thinking" (http://www.cardiff.ac.uk/lean/principles/)

[15] http://agilemanifesto.org/history.html

[16] Sutherland, Jeff. "Agile Principles and Values" (https://msdn.microsoft.com/en-us/library/dd997578(v=vs.120).aspx)