AWS Professional Services – Advisory Practice ::  
**DevOps Transformation Offering in 2017**

v.0.1

**1. Introduction**

AWS customers have many offerings available from ProServe to assist in successfully migrating to Amazon Web Services. For some customers, relieving the operational burdens and costs of managing datacenters is their primary motivation for adopting AWS. Our current approach and offerings are exceedingly capable a delivering this outcome.

For others, migration and operational optimization are the necessary first steps towards unlocking something much more transformative and strategic: reinventing of business through custom-developed, software-driven experiences.

These enterprises want to leverage the agility, scale, and reach of AWS to fuel their ability to become much more responsive to their customers’ needs, competitive threats, and existential disruptions. In today’s digital marketplace, this requires them to ship higher-quality software, faster than ever before, into the hands of their customers. They understand that to be successful going forward, that they must continuously iterate and adapt; rapidly experimenting, learning, and recreating their products in response to what their customer’s experience proves to be most valuable. In the face of this constant change, they must also remain secure and continuously available.

Adopting cloud infrastructure removes the first bottleneck in their journey, but for most enterprises the path forward to enabling the rest of their software-enabled ambitions remains unclear.

**The key ask is for ProServe Advisory to develop a comprehensive “DevOps Transformation Program” offering, to be executed across AWS ProServe.**

This program would build and provide prescriptive guidance, patterns, and tooling to lead aspiring digital enterprises through the transformation of their IT department’s internal operating model, and its integration with the larger business. It will include the necessary people, process, and technology changes – across architecture, development, infrastructure, operations, security, human resources, finance, and, finally, the business itself – required to create and sustain a highly responsive IT organization that truly listens to, and delivers on, its customers’ needs.

**2. Desired Outcomes for Customers & AWS**

The DevOps approach to transformation is designed to extend DevOps’ fundamental cultural focuses of: 1) optimizing end-to-end outcomes; 2) enabling and generating feedback, and; 3) valuing an environment of continuous experimentation and learning; across the IT organization, to reach the following outcomes:

*Customer:*

1. A customer-obsessed operating model for IT that is able to be more successful in responding to, and delivering on, customer needs.
2. An efficient organizational structure for reaching the business’ desired outcomes, based on highly-accountable, balanced, cross-functional product teams and product portfolios.
3. An iterative prioritization and funding model that is tightly coupled with overall business priorities, dependent on customer feedback and results rather than poorly understood requirements and conjecture.
4. A high-quality, high-velocity software deliver capability that rapidly and safely prototypes, enhances, and delivers products based on continuous feedback using continuous integration, continuous delivery, test-driven development, and A/B testing practices.
5. An enterprise architecture that facilitates safe, frequent, and autonomous development and release cycles by small, full-lifecycle teams by decoupling complex workloads into more purposeful microservices and functions that own their own data.
6. An empowered and engaged engineering community that promotes ownership, information sharing, continuous learning and experimentation, and building a culture of safety and trust.

*AWS:*

1. Adoption of more modern architectures and technology stacks by the customer, driving expanded consumption of “sticky” AWS KRADL services and Code\* tools
2. Perception in the market of being not just an infrastructure provider, but a partner that enables business reinvention to compete and win against digital disruptors.
3. Creation of a flywheel effect that allows the customer achieve more with AWS’ approach than they would have on their own; driving more of the customer’s workloads towards this model, and attracts more customers in the process.

**3. Tenets**

**Think Big:** The goal is to create a product-based organization that embraces DevOps culture and practices across the entire IT organization to delivery products that delight customers (internal and external customers).

1. **Outcome-based**: Products exist deliver business customer outcomes, not personal needs (Customer-Obsessed)
2. **Iterative** (Are Right, A Lot)
3. **Optimize for Responsiveness** (not cost) (Bias for Action)
4. **Empowerment/Autonomy/Accountability** (Ownership)
5. **Pervasive Feedback, Learning, & Sharing** (Learn and be Curious)
6. **Safety & Trust** (to Invent and Simplify)

*For more information on DevOps, please see the appendix.*

**4. Delivery Components**

**1. Implement a Product-Based Operating Model**

* Establish a product-oriented operating model as go-forward strategy
* Create a product taxonomy to classify aggregate business services from their component products and outcomes
* Assess and refactor current operating model and IT value streams
* Create “as-is” and “to-be” systems architecture to map business outcomes to applications and technologies

**2. Align Funding and Prioritization with Business**

* Enable business and IT executives to effectively prioritize initiatives based on business priorities and technology vision
* Define an ongoing, iterative prioritization and funding process to support the product operating model

**3. Align Organizational Structure and Talent**

* Leverage product taxonomy and architecture to identify an appropriate product-based organizational structure
* Define labor-model and balanced team-structure for product org
* Map existing talent into product-based, balanced teams where possible
* Partner with HR to help manage OCM, fill talent gaps, and enhance hiring

**4. Scale Modern Engineering Practices**

* Establish core engineering practices to support product model: Agile/XP, CI/CD, Microservices & APIs, Infrastructure as Code, Performance and Metrics, & Cloud
* Build an immersive learning center where teams can be coached on above practices through long-duration “real-work” development

**5. Foster a Modern Engineering Culture**

* Develop forums and tooling to promote a high-trust, collaborative culture than encourages continuous learning and experimentation: Internal conferences & competitions, demos, blameless post-mortems, social coding, internal open-source and code re-use, ChatOps, and many others
* Remove sources of internal friction between dev, ops, and security through self-service automation: proxies, firewalls, permissions, etc..

**6. Develop & Enable Cloud Platform Excellence**

* Develop standard and reusable technology stacks, patterns, and automation to deliver cloud native platforms and applications.
* Develop a cloud platform product team (CCOE) to implement integrations between the cloud and corporate data centers, cloud platform standards, and to coach others through cloud adoption

**7. Measure and Drive Results**

* Incorporate key success metrics and goals into career performance reviews
* Assess, baseline, improve, and report on DevOps outcomes
* Instrument, assess, baseline, improve, and report on key DevOps engineering practices and platforms
* Identify key stakeholders and establish review cadence to assess progress
* Establish org-wide communication plan for sharing updates and progress

**Appendix A**

**Defining “DevOps”**

DevOps | *dəˈv • äps* |

*a cultural and professional movement, focused on how to build and operate high velocity organizations, born from the experience of its practitioners*

[*https://learn.chef.io/skills/foundations-of-devops/*](https://learn.chef.io/skills/foundations-of-devops/)

**Introduction & Background**

**DevOps is a cultural and professional movement** that emerged from “web scale” startups that were forced to deploy and operate systems with high-velocity and ridiculous scale, with not enough people, in a way that was safe and humane to engineers. They did this while moving their fledgling businesses forward in a way that leveraged software and digital business models to continuously add value for their customers, rapidly expanding their marketshare in the process.

**DevOps is a culture and a value system**. DevOps culture values achieving end-to-end “outcomes” rather than commoditized and undifferentiated “activities”. These outcomes are achieved by fostering and instilling a deep sense of trust, purpose, empathy, speed, quality, feedback, and continuous experimentation and learning all at the same time, across a single organization. These qualities can be developed within a single team, or expanded to empower and ignite an entire IT organization. "DevOps", in this sense, can encompass not only “Dev” and “Ops”, but also QA, Infrastructure, Security, and “The Business” itself.

**DevOps is often confused and mistakenly identified by its implementation details.** DevOps is not Infrastructure as Code. It’s not Docker or CloudFormation. It’s not Continuous Delivery. It’s not putting developers and operations staff on the same team. It’s not Agile software development. It’s not developers doing operations.

**These technologies and approaches may signal that a DevOps culture is operating within an organization, as they support the outcomes DevOps seeks to achieve, but they are not the goals of DevOps in and of themselves.** You can be doing all of these things, and still have teams working in silos, unaware of their work is affecting others' work, or how it connects to, or supports, the business. Conversely, DevOps culture and practices can be leveraged with success even in organizations that still organize and perform their individual work activities in functional silos.

**Generally Accepted Definitions of DevOps**

Due to DevOps’ origins outside of Amazon, and its central theme of DevOps being a global community, it is therefore valuable to map Amazon constructs - like the Cloud Adoption Framework - to external definitions and explanations of DevOps if we are going to bring DevOps practices to our customers.

**There are two primary and accepted definitions of DevOps in the global DevOps community:**

1. The Three Ways
2. CALMS

*The Three Ways*

The Three Ways was first documented as part of "[The Phoenix Project](https://www.amazon.com/dp/0988262509)" by Gene Kim, Kevin Behr, and George Spafford. "[The DevOps Handbook](https://www.amazon.com/dp/1942788002)" by Gene Kim, Jez Humble, Patrick Debois, and John Willis expands on The Phoenix Project and discusses how to drive DevOps culture, practices, and outcomes within an enterprise.

**The First Way**: Develop Systems Thinking

Systems thinking is where the DevOps concepts and values of empathy, value-streams, balanced (cross-functional: biz, dev, ops, security) teams, lean thinking, and "full-stack" or "full-lifecycle" engineering and operations practices come from. It's also central to the theme of working across the functional silos of Development, Operations, Infrastructure, QA, Security, and "The Business". Working backwards from the customer - whoever that customer may be - is a very effective way to encourage systems thinking.

**Additional qualities of systems thinking include:**

* Optimizing performance of the entire system vs. a specific silo of work or department
* Removing waste from the IT delivery value-stream
* Never passing a known defect downstream
* Always seeking to increase (single-piece) flow

**The Second Way:** Amplify Feedback Loops

Amplifying feedback loops is all about making sure we are doing the right thing for both the customer (whoever that may be, internal or external) at all times. Doing this successfully requires us to know - as quickly as possible - when we are on track with this goal, and when we are not.

*Amplifying feedback loops can take many forms:*

* Infrastructure as Code, telemetry, metrics, monitoring, logging, performance engineering, visualization and dashboarding, A/B testing, canary deployment, and blue-green deployment are all methods use to amplify feedback by speeding up the time it takes to deploy infrastructure and applications, while minimizing the risk of these deployments, and then to learning if that infrastructure and associated applications are meeting IT's expectations and adding value to customers.
* Continuous, automated testing - across functional, integration, security, availability, and performance domains, is how we ensure we can safely rollout new features and fixes, so that we can more quickly collect feedback from customers.
* Agile software development - XP, Scrum, etc... - to break down work into small meaningful deliverables that can be effectively tested, deployed, and learned from... so that the business can change course if necessary.
* Microservices architectures and empowered, cross-functional, two-pizza teams that manage the application’s full lifecycle (build-it-run-it) are ultimately about amplifying a team’s ability quickly create hypotheses and tests, via software, and then rapidly collect feedback on both. This is hard, if not impossible, with a large, monolithic codebase; where a change in one part of the code may have unpredictable, cascading impact across the entire application. Coordinating changes across software teams engineering and supporting these monoliths is also challenging, which leads to exhaustive CAB meetings and infrequent release cycles… all of which impedes our ability to hypothesize, test, and collect meaningful feedback on a regular basis, which minimizes how responsive we can be to customers. Having microservices with APIs backed by strong contracts allows our software lifecycle to move more quickly and independently, without worrying about disrupting the larger system. Small, cross-functional teams minimize the coordination and communication costs that bog down larger systems.

*Other aspects include:*

* Shortening and amplifying feedback loops so necessary corrections can be continually made
* Understanding and responding to all customers, both internal and external
* Embedding knowledge where necessary

**The Third Way:** Foster a Culture of Continual Experimentation and Learning

This may be the most important Way, as DevOps’ drive for continuous experimentation, learning, and sharing is what fuels the need for empathy and feedback in the previous two ways. You must foster a culture of continual experimentation and learning for your organization to find value from understanding and improving a business outcome end-to-end, not just individual activities within a specific silo; to value continuously deploying small changes into production, and then empirically learning through amplified feedback loops whether the change actually added the intended value.

This is also where we develop an organization's sense of purpose, trust, and sharing. This can be done by doing things like converting "root cause analysis" meetings to "blameless post-mortems", so we can learn from our failures, instead of blaming and holding people accountable in a punitive way. Culture can be developed by doing public demos as part of end-of-sprint retrospectives, holding internal tech conferences where people can share their work and experiences, and increasing collaboration through ChatOps tools, open workspaces, and internal hack-a-thons.

**Other key aspects include:**

* Creating a culture that fosters continual experimentation, taking risks, and learning from failure
* Understanding that repetition and practice is the prerequisite to mastery.

**CALMS**

CALMS is an acronym that stands for Culture, Automation, Lean, Metrics, and Sharing. It was originally “CAMS” when introduced by Damon Edwards in 2010, and Jez Humble later added Lean into the equation.

CALMS is a way talking about HOW to implement the What and Why that are embodied in Gene Kim's "Three Ways". As such, it is fairly prescriptive:

**Culture**

* Own the change to drive collaboration and communication
* Examples: Collaboration, Continuous Learning, & Empathy

**Automation**

* Remove manual steps from your value chain
* Examples: Continuous Delivery, Infrastructure as Code, Infrastructure and Platform as a Service

**Lean**

* Use lean principles to reduce cycle time
* Examples: Reduce handoffs and work queues using value-stream mapping + automation

**Metrics**

* Measure everything and use data to refine cycles
* Examples: Uptime, Performance, Response Time, Delivery Time, Quality, etc…

**Sharing**

* Share experiences - successes and failures to enable others to learn
* Examples: Demos, Blameless Post-Mortems, Internal and External Tech Conferences