



DevOps Primer: Case Studies and Best Practices from Across Government

ACT-IAC EMERGING TECHNOLOGY COMMUNITY OF INTEREST
DevOps Working Group

Date Released: February 19, 2020

Synopsis

DevOps is both a new way of thinking and a new way of working. It is reshaping how organizations innovate and quickly deliver positive business outcomes. However, transitioning to DevOps requires an essential shift in attitudes and approach. ACT-IAC developed this primer to be a guide to support the Federal government in its understanding and application of DevOps to support its mission.

This paper provides:

- Case studies highlighting where agencies are in their DevOps adoption and implementation journey. It is a collection of lessons learned and ideas for how committed IT professionals can persist through the challenges and make progress.
- A Maturity Model with stages of advancement which organizations can use to move toward a state of continuous improvement.
- Recommendations from government and industry leaders to advance the DevOps journey.

This page is intentionally blank.

American Council for Technology-Industry Advisory Council (ACT-IAC)

The American Council for Technology (ACT) is a non-profit educational organization established to create a more effective and innovative government. ACT-IAC provides a unique, objective, and trusted forum where government and industry executives are working together to improve public services and agency operations through the use of technology. ACT-IAC contributes to better communication between government and industry, collaborative and innovative problem solving, and a more professional and qualified workforce.

The information, conclusions, and recommendations contained in this publication were produced by volunteers from government and industry who share the ACT-IAC vision of a more effective and innovative government. ACT-IAC volunteers represent a wide diversity of organizations (public and private) and functions. These volunteers use the ACT-IAC collaborative process, refined over thirty years of experience, to produce outcomes that are consensus-based. The findings and recommendations contained in this report are based on consensus and do not represent the views of any particular individual or organization.

To maintain the objectivity and integrity of its collaborative process, ACT-IAC does not accept government funding.

ACT-IAC welcomes the participation of all public and private organizations committed to improving the delivery of public services through the effective and efficient use of IT. For additional information, visit the ACT-IAC website at www.actiac.org.

Emerging Technology of Community of Interest

The ACT-IAC DevOps Working Group was created to identify and explore the culture, practice, and tools needed to increase the government's ability to deliver applications and services at a faster pace than using traditional software development and infrastructure management processes. This speed will enable the government to better serve their customers and fulfill their mission. The ACT-IAC Emerging Technology Community of Interest mission is to provide an energetic, collaborative consortium comprised of leading practitioners in data science, technology, and research, engaged with industry, academia, and public officials and executives focused on emerging and leading technologies which transform public sector capabilities.

Disclaimer

This document has been prepared to contribute to a more effective, efficient, and innovative government. The information contained in this report is the result of a collaborative process in which a number of individuals participated. This document does not – nor is it intended to – endorse or recommend any specific technology, product, or vendor. Moreover, the views expressed in this document do not necessarily represent the official views of the individuals and organizations that participated in its development. Every effort has been made to present accurate and reliable information in this report. However, ACT-IAC assumes no responsibility for consequences resulting from the use of the information herein.

Copyright

©American Council for Technology, 2020. This document may not be quoted, reproduced and/or distributed unless credit is given to the American Council for Technology-Industry Advisory Council.

Further Information

For further information, contact the American Council for Technology-Industry Advisory Council at (703) 208-4800 or www.actiac.org.

CONTENTS

INTRODUCTION	6
What is Agile?	7
What is DevOps in Government?	8
DevOps Technical Overview	9
High-Level Approach	9
DevOps Maturity Model	11
Government Case Studies	13
U.S. Patent and Trademark Office	14
U.S. Citizenship and Immigration Services	16
Internal Revenue Service (IRS)	18
Smithsonian Institution: National Museum of African American History & Culture	21
National Park Service	23
National Science Foundation	25
Shared Challenges and Tactics	27
Procurement	27
Shared Challenges	27
Tactics	27
Culture Change	28
Shared Challenges	28
Tactics	29
Communications	30
Shared Challenges	30
Tactics	31
CONCLUSION AND RECOMMENDATIONS	32
GLOSSARY	33
ACKNOWLEDGEMENT	34
Authors and Affiliations	34
REFERENCES	35

INTRODUCTION – WHAT IS DEVOPS?

What is DevOps? An approach? A tool? A role? Many government IT leaders find themselves being pushed towards DevOps but have not had a chance to truly understand its benefits, or how to best leverage it as a tool to drive IT value optimization within their agency.

DevOps is the combination of cultural philosophies, practices, and tools that increases an organization's ability to deliver IT applications and services at high velocity: evolving and improving products at a faster pace than organizations using traditional software development and infrastructure management processes. "This speed enables organizations to better serve their customers and compete more effectively in the market."¹

When executed effectively, it can lead to many of the following benefits:

- Rapid development pipeline - With continuous integration and continuous delivery (CI/CD) and a robust test suite, code changes can be pushed and auto deployed to development/testing/production delivering incremental enhancements as soon as they are completed and in a manner that reduces regression errors.
- Rapid deployment pipeline - If code is committed to a project with a robust suite of integration, regression, and unit tests, there is minimal risk when auto deploying those changes to dev, test, and prod environments as soon as the tests pass.
- Reduced Feedback loops - Because of the constant improvements being implemented, users feel comfortable requesting enhancements, and providing feedback based on their confidence in the application's stability.
- Rapid disaster recovery - When true CI/CD is implemented, there is a robust change history automatically integrated into the source control repository. This provides a history of application states allows for easy recovery and regression if an enhancement has unintended consequences that were not picked up by the test framework.

Through a survey (see Appendix) and hundreds of conversations about what is working and what is not, the ACT-IAC DevOps Working Group shares their insights on DevOps and "what is next" for federal agencies. This is intended to be a guide and not a definitive step-by-step playbook. With DevOps, there is not one right answer. Instead, it requires an essential shift in the attitudes and approach of Federal leaders and staff alike.

In this paper, you will find:

- A maturity model with stages of advancement which organizations can use to move towards a state of continuous improvement.
- Case studies highlighting where agencies are in their DevOps adoption and implementation journey. It is a collection of lessons learned and ideas for how committed IT professions can persist through the challenges and make progress.
- Recommendations from government and industry leaders to advance the DevOps journey.

In many agencies, DevOps is still a small pilot effort on a handful of applications. It usually lacks visibility and struggles to the attention and resources devoted to larger initiatives. Agencies face challenges rooted in the lack of true executive understanding, buy-in, and support. As a result, many leaders and staff devoted to DevOps struggle to define and then meet expectations (both their own and their leaders and appropriators.) IT leaders are taking a more proactive approach to understand the negative forces and barriers that interfere with their ability to fully deliver on the promise of increased speed, quality, and security of DevOps.

This primer was developed for them. It is for IT leaders struggling to gain traction and make the kind of progress they know is possible. Again, this is not a precise prescription. Instead, it is a set of case studies highlighting where others are in their DevOps adoption and implementation journey. It is a collection of lessons learned and ideas for how committed IT professions can persist through the challenges and make progress.

Finally, DevOps also is not just an “IT” thing. To be successful, business or programmatic organizations under the same umbrella mission should understand and adopt DevOps concepts as well. It is the responsibility of IT to explain the benefits and take the lead on setting new, collaborative approaches to defining requirements and delivering products and services, but this is not an endeavor that can be completed in a silo.

Agile or DevOps?

Agile and DevOps are two modern software concepts often employed by many organizations, but there is often confusion or overlap between how they are described. Agile is a software development methodology, while DevOps is an organizational philosophy around how work gets done.

Oxford dictionary defines agile as “able to move quickly and easily”².

	Waterfall	Agile	DevOps
Basic philosophy	Systems are fully predictable and can be specified in advance. Assumes business needs remain broadly similar throughout project. Adjust schedule to preserve scope	Integrate business, dev and QA for rapid delivery of software. Iterative 'sprint' cycles. Assumes priority of business needs may change. Adjust scope to preserve schedule	Cross-functional teams utilize automation to enable continuous deployment of change. Constant feedback loop. Adjust scope to preserve schedule
Documentation level	Comprehensive	Light	Light
Automation level	Low	Varied	High
Delivery of value	Slow – only at major milestones (3-6 months)	Rapid (daily/weekly)	Continuous
Business ownership of project?	No (typical)	Yes	Yes
Response to new business needs (flexible requirements)	Extremely limited due to detailed specification	Responsive – iterative delivery enables prioritization	Highly responsive – cross-functional teams define business needs more precisely
Collaboration	Low – teams operate in functional silos	Improved – business is highly engaged, short dev cycles	High – all stakeholders involved from project start
Quality	Low – issues not identified until testing phase.	Improved – issues identified after every 'sprint'	High – automated unit testing during development
Risk	Increases as project progresses	Decreases as project progresses	Decreases as project progresses
Customer feedback	Infrequent - at project completion	Frequent – after every sprint	Continuous

Figure 1: Comparing Waterfall, Agile, and DevOps (Source: Dzone.com)³

In relation to project management for software development, agile is characterized by the division of tasks into short phases of work and frequent reassessment and adaptation of plans.⁴ It is an evolution of software development designed to overcome the limitations of “Waterfall”.

DevOps was not designed to replace or improve agile, but rather evolved separately as a philosophy under which the business function / teams, development teams, and the operations team collaborate on a continuous basis to make sure that IT solutions are available to business on time and that they run without disruption. It calls for cultural change, organization structure change, and automation, with an emphasis on people, process, and tools; allowing for greater collaboration and understanding across stakeholders.

Staying true to the dictionary definition, agile offers shorter development cycle, while DevOps supports agile’s shorter release cycles and the overall iterative approach through increased collaboration, visibility, and automation across all stages of software development. When leveraged in tandem, agile can bring customer voice and business alignment to development, while serving as a motivator for DevOps culture, while DevOps can improve the speed and quality of delivery.

What is DevOps in Government?

DevOps* is the move towards a collaborative and communicative mindset and culture with streamlined practices by bringing the development and operations groups together to work as one team. DevOps is a continuous journey with no defined end point.

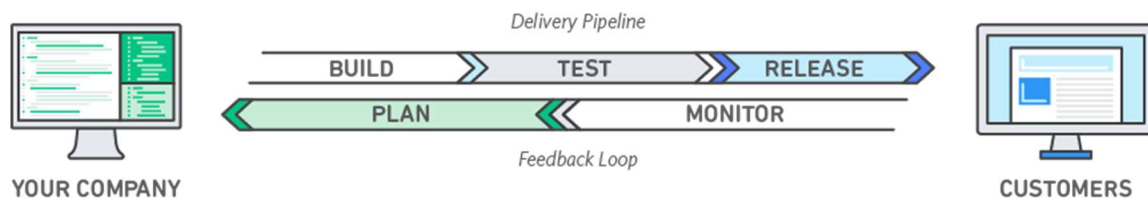


Figure 2: DevOps Model (source: Amazon Web Services)⁵

Today, many Federal agencies are several years into their DevOps journey and wondering: Is this it? Are we doing it right? Why is this so difficult?

IT leaders are taking a more proactive approach to understanding the negative forces and barriers that interfere with their ability to fully deliver on the promise of increase speed, quality, and security of DevOps.

* Some agencies opt to call their programs DevSecOps to emphasize the importance and role of security in their efforts. Others believe security is “baked in” and choose to use the more ubiquitous DevOps term.

DEVOPS TECHNICAL OVERVIEW

High-Level Approach

In October 2016, ACT-IAC published a report entitled “LEGACY SYSTEM MODERNIZATION: Addressing Challenges on the Path to Success” and defined DevOps as:

“...strategy [that] incorporates intelligent coding and continuous maintenance into the legacy system while continuing to use the existing code and platform. A DevOps framework supporting modernization extends Agile development by leveraging automated tools to enhance continuous integration throughout the lifecycle. This creates a legacy modernization process for a “continuous delivery pipeline” to plan, define, develop, test, release, and monitor releases, leveraging automation throughout the legacy modernization lifecycle. Developer-friendly Integrated Developer Environments (IDEs) and tools are used, which partially address the reduced availability of skilled resources for legacy languages. This solution introduces a collaborative working style between the development and operations teams. Changes are deployed faster and on an as-needed basis. Continuous delivery enables automated deployment and verification of an application across a set of environments. Application characteristics suitable to DevOps/Sustainment: Applications that functionally meet (or mostly meet) the business need. Agency is looking for ways to ease future enhancements and reduce release schedules while retaining the legacy language or platform.”⁶

That is one definition of DevOps and there are many other versions. At the same time, almost everyone will agree that DevOps can be accomplished by implementing a robust Continuous Integration / Continuous Deployment (CI/CD) pipeline. A CI/CD pipeline is where environments have different stages such as Development (Dev), Integration (Int), Quality Assurance (QA), User Acceptance Test (UAT), Staging / Pre Production (Stage/PP), and Production (Prod). The CI/CD pipeline allows manual processes to be automated. Such a pipeline allows individuals to achieve high quality code with the flexibility to deploy/rollback without impeding other’s work as well as numerous deployments.

The tools (including some open source options) mentioned below are examples of what are currently available and do not imply preference or endorsement from ACT-IAC nor the Working Group.

Creating a CI/CD pipeline usually follows this process:

CI/CD tool – staying consistent with being agnostic, using your favorite search engine, search the terms – ci cd tools. You will come across many results (including top 5, 7, 10, etc.) However, you will find a few names that are very common within the DevOps community (e.g., Jenkins, Travis CI, CircleCI, GitLab CI, to name a few.) These are automation tools that, when plugged in with other tools in the pipeline, allow you to

automate various manual aspects of build, testing, deployment, etc. Imagine an orchestra conductor.

Version control tool – this is a repository (repo) of your code. Once code is developed using one of many programming languages (Java, Ruby, Python, to name a few), the code needs to be stored at a common place - a repo. These repo tools allow you to version your code. When multiple individuals are working on a file, these tools automatically version the file. Again, search for version control tools and you will come across many results. Some of the more common ones in use are CVS, SVN, GIT, and Mercurial. Most of these tools will allow multiple individuals to work on same file (code file) concurrently, and once the file is checked-in by the team member, the tool automatically assigns a version number.

Build tool – code that is written to develop application must be compiled (with few exceptions – Javascript, PHP or some other Markup Languages do not require compilation). Code is compiled (or bundled together) to ‘build’ an executable file (e.g., for Windows users, this is a .exe file). This process of compiling and ‘building’ is done by a Build tool. A Build tool is selected depending on the programming language chosen to write the code. Some common build tools are Maven, ANT, Gradle, and Rake.

Server – this is where your compiled code is hosted. Imagine this as a place where all of your software runs (e.g. for Windows Operating System (OS), you run your notepad, office, or browser applications on the OS). An application server does the same job, but for your custom software code. There are many open source application servers available today: Apache TomCat, Django, Rails, and Node.js to name a few.

Tests – Testing is often the albatross of software development projects. Testing tends to slow the development lifecycle or delay the software deployment. The purpose of injecting testing in the CI/CD pipeline is to avoid such occurrences by speeding up tests as well as shifting the tests to left. This is also where you will inject security. “Shift the tests to left?” means testing small components early and incrementally and automating. One of the most common ways to test your components are by wrapping your feature code with Unit tests. Including Security Analysis Security Testing (SAST) and Dynamic Analysis Security Testing (DAST) can integrate thorough security checks in the development process. There are many open source test frameworks and tools available: Junit, Selenium, Cucumber, Watir, Appium, TestNG, JMeter, Gatling, Locust, SonarQube, Bandit, LGTM, and more.

With that process, the CI/CD DevOps pipeline is ready. This approach is meant for organizations /teams to get started with DevOps. Using the suggested open-source tools enable this approach to scale withing the governance guiderails.

DevOps Maturity Model

To support the adoption and implementation of DevOps, the Working Group developed a Maturity Model - a framework organizations can use to examine their current level of maturity and develop a path forward. It shows an evolutionary process of advancement that leads to a state of continuous improvement. Each agency will find its own unique path and may opt to focus on a subset of these initiatives instead of doing everything at once.

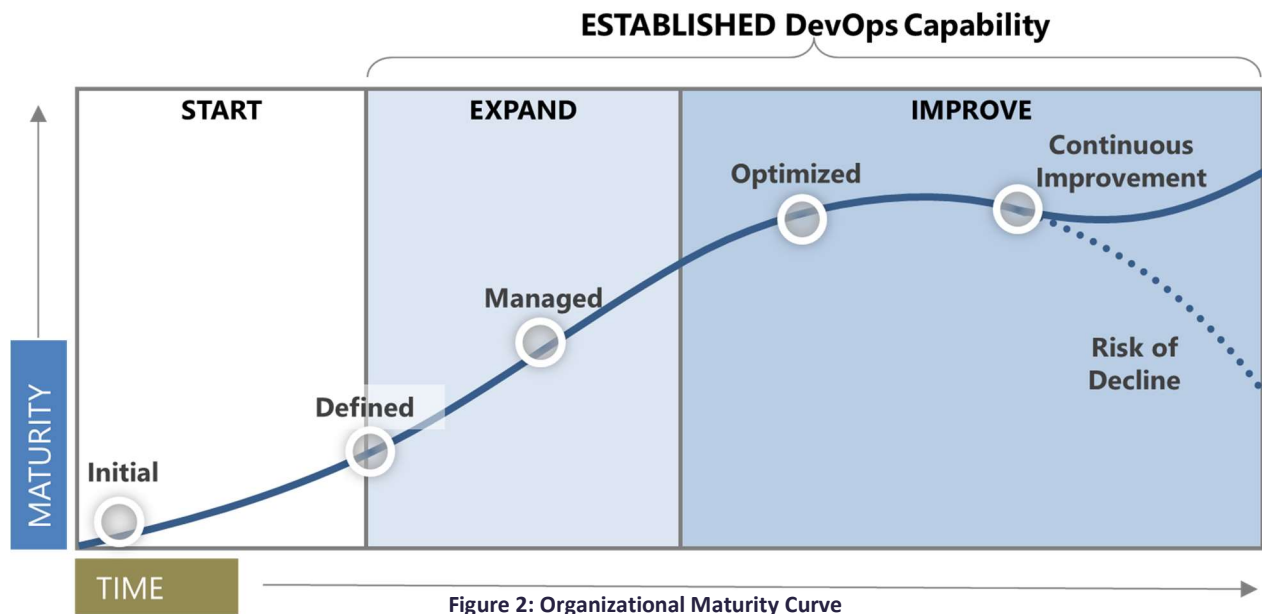


Figure 2: Organizational Maturity Curve

Initiative	Initial	Defined	Managed	Optimized	Continuous Improvement
Build, Deploy, & Release	Manual deployments	<ul style="list-style-type: none"> Some automated deployment scripts Environment tailoring required Few enterprise-level tools 	<ul style="list-style-type: none"> Automated deployments into environments 	<ul style="list-style-type: none"> Orchestrated deployments 	Zero-touch, zero downtime deployments
Testing	Mostly manual test scripts	<ul style="list-style-type: none"> Test strategy defined Automated unit tests Separate test environment 	<ul style="list-style-type: none"> Automate integration testing and database testing for select projects Static code analysis 	<ul style="list-style-type: none"> Fully automated acceptance tests Automated performance tests 	Tests run as a function of continuous monitoring

Initiative	Initial	Defined	Managed	Optimized	Continuous Improvement
Security	Security bolted on at end of development lifecycle	<ul style="list-style-type: none"> Ensure policies and tools developed 	<ul style="list-style-type: none"> Security moved up in development lifecycle for earlier weakness detection and mitigation 	<ul style="list-style-type: none"> Automate security requirements /security control testing Defects identified and immediately fixed 	Security built-in from beginning
Infrastructure	Ad hoc environments, little consistency	<ul style="list-style-type: none"> Establish enterprise code standards Environments manually created 	<ul style="list-style-type: none"> Automated creation of environments All follow common set of blueprints 	<ul style="list-style-type: none"> Enable Cloud processes and capabilities to improve infrastructure use 	Utility-based computing that leverages auto-scaling
Delivery methodology	Poorly defined scope, ad hoc change requests	<ul style="list-style-type: none"> Releases take a long time, delay “business” value 	<ul style="list-style-type: none"> Release cadenced defined Requirements are stable Use business metrics to assess progress and recommend improvements 	<ul style="list-style-type: none"> Release on demand, time-boxed to meet business need 	Continuous deployments further innovation
Culture and Governance	Multiple layers, hierarchical decision-making	<ul style="list-style-type: none"> Culture strategy developed and behavioral changes identified Communities of Practice established 	<ul style="list-style-type: none"> Build buy-in, support, and executive confidence DevOps decision-making process 	<ul style="list-style-type: none"> Blame-free culture embedded in governance Increase range of distributed decisions Organize Cross/Up Skill training strategies to socialize best practices 	Distributed decision-making

GOVERNMENT CASE STUDIES

The Working Group reached out to government agencies about their experience with DevOps. The following case studies were prepared by participating agencies. In their own words, they describe the goals, benefits, approach, lessons learned, and next steps that are specific to their unique missions and cultures. These case studies help highlight possibilities and pitfalls to be avoided.


Each agency scored themselves against the maturity model to help the reader understand the current state of their DevOps journey. Each case study is presented in a two to three-page format. It is recommended that you read these case studies and match them with the requirements matching your organization's mission needs.

Note: An "x" in two phases of one initiative in the maturity model indicates portions of the agency's program fall in both.

Agencies who participated include:


- U.S. Patent and Trademark Office
- U.S. Citizenship and Immigration Services
- Internal Revenue Service
- National Museum of African American History & Culture (A Smithsonian Institution)
- National Park Service
- National Science Foundation

U.S. Patent and Trademark Office

	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release				X	
	Testing			X		
	Security				X	
	Infrastructure			X		
	Delivery methodology			X	X	
	Culture and Governance			X		
Problem Statement and Goals	<p>Our overarching goals were to have all next generation portfolios adhere to Agile methodology. We set out to improve alliance between product owners, developers, testers, cybersecurity, and infrastructure. We've largely reached those goals and have seen the following benefits:</p> <ul style="list-style-type: none"> • Increase in embracing DevOps culture to streamline product deployment • More rapidly delivering new features to production • Minimizing manual involvement in building and releasing a software change, and • Increasing visibility into key metrics to determine code promotion to the next environment. 					
Approach	<p>PTO started DevOps in 2012. Our approach was grounded in four objectives: make incremental changes, focus on collaboration, remove organizational barriers whenever possible, and continuously improve. These helped shape the specific actions taken in each of six completed phases.</p> <p>Phase 1: We established our initial operating capability by implementing an enterprise software versioning and source code management platform. We used open source solutions for a build platform that automated the deployment process. In addition, we provided coaching for migrating existing products to the new platform.</p> <p>Phase 2: We focused on improving build automation by implementing a Build Once Deploy Anywhere (BODA) framework. We defined critical metrics for all development teams and published their results on Sonar dashboards for transparency.</p> <p>Phase 3: We fully automated testing in all environments. We focused on improving timely bug detection (fast and reusable) and improving test coverage. When Phase 2 was complete, we could run most test plans 24x7 unattended.</p> <p>Phase 4: We fully automated the deployment process and released more frequently with repeatable deployments.</p> <p>Phase 5: We focused on establishing quality gates by continuously testing every release candidate for quality and security. We concentrated on detecting and remediating defects and critical security vulnerabilities earlier during development. We implemented a one-click automated build and deployment process.</p> <p>Phase 6: Most recently, we have been working on continuous improvements. We are optimizing the platform to capitalize on emerging technologies. We are facilitating and extending enterprise</p>					


	<p>capabilities – Software Asset Management (SAM). Additionally, we are optimizing the deployment process by fully automating deployments, including full tech-stack.</p>
<p>Lessons Learned</p>	<p>Loosely-Coupled Architecture: When we started onboarding teams to the CICM platform, a Build Once Deploy Anywhere (BODA) framework was defined which was ideal for loosely coupled architectures. We quickly learned that teams had to refactor to adhere to the BODA framework, which caused lots of work for teams to refactor their code. We have changed the approach now and work closely with the Enterprise Architecture, so that teams starting development can develop non-monolithic applications that can be managed efficiently.</p> <p>Tools: Earlier there were manual steps in even building a Production-Ready release candidate. We have introduced an automated deployment CICD pipeline for all teams that eliminates manual steps, delays, errors and rework.</p> <p>Process: We did not have set of processes defined when the platform was stood up. We learned that to effectively manage a repeatable build & deploy process, we had to invest on right processes & procedures up front. The Automated Deployment Pipeline framework & DevOps Policy & Procedures documents were created to provide guidance for USPTO software products teams to fully utilize DevOps & the Automated Deployment Pipeline.</p> <p>From a management perspective, there are three key things.</p> <ol style="list-style-type: none"> 1. If the decision has been made by an agency to use a CICD pipeline, then all new projects must be on boarded from the “get go” instead of trying to retrofit them later. Using this approach saves us all a lot of rework at a later time. It provides the necessary automation from the start and more comprehensive code coverage. 2. Dissolving the organizational silos is critical. All divisions, office, branches, etc. must be equally tasked with the goal of meeting DevOps objectives. It can’t just be one organization’s goal and not the others. 3. Engaging the business side early on in adopting the agile philosophy is essential. In the USPTO the Patents and Trademarks divisions are our customers. For all products we’re developing, we need a business owner. Business unit representatives must be assigned to and aligned to the development project for it to successfully meet their needs.
<p>What’s next?</p>	<p>One of our biggest goals is to provision virtual machines using our CICD pipeline while making them secure. We used to create servers in every environment by hand. Today, we are creating servers using automated scripts.</p> <p>We want to take that to the next level by building the generic servers and automatically configuring them for the product.</p> <p>We plan to add more tools and continue to shift activities “to the left” or earlier in the process. DevOps is continuous improvement & we are adding metrics, so teams can continuously improve as they build an application.</p> <p>We also want to spread the DevOps concept and philosophy across our organization- beyond the Office of the Chief Information Officer. Other organizations would benefit from DevOps by breaking down the silos and increasing peoples’ ability to work together. DevOps does not just have to be about technology. It’s taking the agility that comes to us by using these methodologies and spreading them across the entire lifecycle—from procurement to the business to system retirement.</p>

U.S. Citizenship and Immigration Services

 U.S. Citizenship and Immigration Services	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release				X	
	Testing			X		
	Security			X		
	Infrastructure				X	
	Delivery methodology			X		
	Culture and Governance				X	
Problem Statement and Goals	<p>A big driver for moving to Agile and DevOps is the pain of not having fully functional code in production that could keep up with our caseload. Before this initiative, USCIS had been spending money but not seeing the immediate value. The conversation eventually focused on the core idea of delivering business value. “If we’re going to spend this money, we should see some value earlier in the project.” Agile and DevOps were specific approaches to address this pain.</p> <p>At the same time, there was a business need for new functionality to handle additional forms and an ever-increasing caseload. As an illustrative example, USCIS was processing 50 cases a month. The organization needed more, and they started setting stretch goals. One of the first goals was to 70 cases per month. The team figured out how to reengineer the process to reach that goal. In 2011, release delivery cycles could take more than 180 days. Over the next 8 years, USCIS decreased the delivery cycle down to deploying once or even multiple times a day. The organizations were better able to scale after that as well as decrease the amount of documentation needed for these releases.</p> <p>USCIS’s journey started in 2011. Key to getting DevOps underway was having the senior leadership guiding and mandating agile and DevOps automated frameworks and practices. Initially, there was a lot of resistance within the organization to move from traditional waterfall and manual processes, but the executive buy-in was critical in changing the mindset.</p>					
Approach	<p>USCIS realized early on that one of their first challenges was not technical at all, and that it was the process for which USCIS procured/contracted for development teams and technology. Traditional contracting did not work in an Agile/DevOps environment because of the lead-time needed, changing requirements, and the contract duration. As buyers, you want to be able to try and test things and then adjust the requirements as you learn more. There was a mindset shift needed in procurement, IT, and on the business side. An innovation in contracting for USCIS was initiating demonstration-based evaluations in the shape of Technical Challenges and Technical Demonstrations.</p> <p>Now when USCIS acquires an agile team, the vendor is required to demonstrate knowledge of agile processes, ability to code, and work on the user stories assigned to them, etc. The written proposal is less important than their visual demonstration of their approach and how they work together.</p>					

	<p>The other important piece early on was engaging US Digital Service (USDS.) USDS was the group coming out of the HHS and the Obamacare marketplace. They started applying those approaches and best practices out to other agencies by way of their US Digital Services Playbook (https://playbook.cio.gov). USCIS, a key beneficiary of their playbook, would not have been as far along without that initial help.</p> <p>After 2013, USCIS began to align internally to a DevOps model to maximize the Agile principles and DevOps practices and remains in place today. One of the bigger influences to organize around Agile/DevOps was leveraging cloud computing. USCIS began implementing a cloud infrastructure to start realizing some of the automation benefits. ELIS (Electronic Immigration System) was the first project to prove out DevOps. Today, there are approximately 20+ systems and applications that utilize DevOps and a CI/CD pipeline to push their code up through various test environments and eventually into production.</p> <p>USCIS is deploying microservice architectures, essentially platform as a service, to manage their infrastructure. Specifically, USCIS is using container technology to develop the applications that they are moving into production. For example, recently, USCIS launched its first form via their eProcessing initiative. This initial form, I-539 Application To Extend/Change Nonimmigrant Status relies on a micro service architecture that leverages an orchestration platform to manage the process.</p> <p>From a communications perspective, development teams’ partner with the business customer, the Product Owner, to gather requirements in regularly scheduled grooming sessions. They sit down with their product owner, gather requirements, and demonstrate functionality to gain approvals and show that they are making progress. USCIS leverages JIRA primarily to manage backlogs and progress, however additional collaboration tools such as LeanKit and AgileCraft are used as well. These transparent collaboration tools allow the product owners to get a sense of what is being accomplished and quickly prioritize or re-prioritize.</p>
<p>Lessons Learned</p>	<p>If we could go back, we would focus more upfront on the governance of Agile and DevOps. Specifically, we would take more time to decide on and document the guiding principles about what teams can and cannot do with a clearly defined validation procedure. Additionally, a well-defined approval process for the build pipeline itself would create more transparency and have helped with the day-to-day management concerns.</p> <p>Second, we believe there is value in deciding as an organization what tool set you want to use. Today, we have a suite of tools that came out of a lot of trial and error.</p> <p>Lastly, we would have built some of our communities of practice in parallel with the DevOps policies we were putting in place. The COPs are getting more mature and we’re getting them up to speed. Two to three years ago, they weren’t as robust or functional as they are today.</p>
<p>What’s next?</p>	<p>Today, we’re implementing more of a “software factory” - a repeatable way to develop software and to monitor or catch the security vulnerabilities earlier in the process. A couple of examples are applications that monitor the code libraries that go into building applications. In the spirit of transparency, we want to monitor them for the security vulnerabilities, monitor how the open sources libraries are used, and monitor system performance in general. Our goal is to implement that system and monitor our artifact repositories to automate these builds and ensure we do not violate the end-user license agreements.</p>



Internal Revenue Service (IRS)

	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release		x	x		
	Testing		x			
	Security			x		
	Infrastructure		x			
	Delivery methodology		x	x		
	Culture and Governance		x			
Problem Statement and Goals	<p>Modernizing aging applications and processes has been a long-time priority for the IRS. IT leaders and professionals across the organization knew that to be successful they would need a new way of working. Further, they knew the culture of the IRS IT organization must be in lockstep on priorities, processes, and deliverables.</p> <p>Over the years, the scope and complexity of IT delivery became increasingly challenging.</p> <ul style="list-style-type: none"> The application build and deployment process is mostly non-repeatable requiring significant time and staff intensive manual processes. Quality and security of code were not consistently assessed early in the development cycle leading to delays in code release, heroic efforts to meet deadlines and, ultimately production incidents/downtime. Build and deployment scripts, where used, were of variable quality and did not always adhere to best practices. [lack of repeatability] The manual delivery process contained many steps, introducing opportunities for human error, ultimately leading to non-repeatable and inconsistent deliveries. [lack of repeatability] Transformation of the culture was needed for new software and fixes to be deployed to Production quickly; individuals and the enterprise needed to be influenced to drive new behaviors to support DevOps. [lack of repeatability] 80% to 85% of current infrastructure was not consistently utilized, along with a lack of standards in applications development and deployment. Server sprawl has resulted in over 12,600 servers with many of those servers underutilized or not used at all. Teams also faced difficulty getting servers provisioned quickly for projects. [lack of repeatability, server drift] 					
Approach	<p>Prior to 2014, some development teams installed CI servers to enforce repeatable consistent compile and quality in build processes, and some operations teams were implementing automated delivery processes. In 2016, a small “Cheetah” team was established to explore options and opportunities to bring DevOps more formally to the IRS. They had an ambitious goal to create a fully automated end-to-end software delivery pipeline to enable IRS IT to deliver quality and secure software in a more timely and cost-effective manner. In 2017, the DevOps organization was established and staffed as a cross-functional program.</p> <p>At the onset, the IRS DevOps team set out to provision new environments, tear down old ones, and reduce lead times for new builds from months to DAYS. As a consequence, the IRS intended to reduce the number of environments needed. The goal was to further lower complexities and cost.</p>					

	<p>DevOps at the IRS continues to expand and evolve. The cross-organization team has begun to see benefits. To date, teams have onboarded more than 70 applications into the Continuous Integration Continuous Delivery (CICD) pipeline and made strides in other areas.</p> <p>Specifically, streamlining governance and approval processes is reducing lags in staff productivity. Automating testing and deployment actions allows for faster feedback and actionable corrections, resulting in higher quality products and additional freed up staff hours. All the realized excess capacity freed from these efficiencies allows more time for staff to address other higher priority deliverables. As an added benefit, the IRS offers a better quality of work-life balance by cutting down night and weekend work for staff.</p> <p>Continuous Integration results in fewer issues and incidents. Integrating Security into the development process from the beginning reduced redundant operations and increased quality and protection of the code. All these integrated tools simplified communications, sharing of information, and collaboration across organizations.</p>
<p>Benefits</p>	<p>The IRS has faced internal and external challenges when trying to address its front-end user experience for the US taxpayer. In the growing technological world of real-time transactions, the IRS knew they needed to be at the edge of innovation to keep pace with the ever-changing environment--but within a cost-sensitive and long-term investment mindset. Old legacy systems left the IRS hindered by manual processes that lacked security and quality controls which led to time lags, increased costs, and missed delivery windows.</p> <p>Across the current portfolio utilizing the DevOps approach (CICD pipeline, automated testing, standard stacks, and containers), the IRS is seeing the beginnings of the following benefits. As additional applications are onboarded, the IRS and the US Taxpayer will see more robust, positive changes such as:</p> <ul style="list-style-type: none"> • More secure and faster delivery of updated, new solutions for taxpayers at reduced operating costs (30 hours of person work down to 4 hours of person work) • Fewer organizational silos making it easier to bring the best available resources together and makes IRS a more effective, desirable place to work • Increased flexibility and enhanced transparency to manage risks or issues • Unified business development & infrastructure teams able to provide business value and continuous delivery • Better quality through improving change management and enhanced testing • To date - Over 16,000 hours of manual/person work have been automated by the DevOps pipeline annually. • Adding continuous integration and delivery to legacy code migration processes (modernization) <p>The DevOps “Target State” will be nimble, transparent and adaptable to environmental shifts to deliver seamless updates without interruption to end-users.</p>
<p>Lessons Learned</p>	<p>DevOps requires building new organizational behaviors. The IRS continues to focus on how to be more open to risks and trying new things while simultaneously building and keeping leadership support. We know a high-performing DevOps organization requires more flexibility and room to make mistakes. This is difficult within the IRS where there is so little room for error because of the high potential impact on the taxpayer.</p>


	<p>Another key lesson is focusing on building cross- organization bonds, communication lines, and standard processes. DevOps within the IRS crosses multiple well-established divisions. Working across organizations takes time and dedicated, persistent effort. Without it, DevOps slips down the priority list as organizations tend to more pressing “internal” business.</p> <p>DevOps isn’t all about tools. Tools are critical but IRS is now increasing our attention on processes and culture change. We will benefit from revising, standardizing, or replacing outdated policies and procedures.</p>
<p>What’s next?</p>	<p>In prior years, we did the foundational work on smaller projects. Going forward, we intend to help accelerate IRS modernization by freeing up much-needed resources to be reinvested in other priorities.</p> <p>Key Goals for FY20</p> <ol style="list-style-type: none"> 1. Continue to support the IRS Modernization Plan 2. Deliver additional projects to components of the DevOps pipeline. Expanding current technical capabilities of DevOps pipeline to further integrate containers, and standard stacks 3. Define and execute plan to expand DevOps benefits and footprint to larger programs <p>Communications and employee engagement will be the drivers of the cultural change that drives the DevOps Practice. We will garner executive-level buy-in and support for DevOps by more effectively demonstrating progress and accomplishments. Cultural changes must also be made at the executive (SES) level as technical experience with DevOps and automation (technical excellence) will be needed to drive this work (See SaFE documentation)</p> <p>Awareness and understanding across lower levels of the stakeholder groups is integral to success. IRS can accomplish this through a series of controlled and scheduled communications efforts across different media channels, stakeholder forums, and targeted messaging to specific groups. This initiative will allow those consumers of the information to continue to drive the successful operations of the DevOps Practice across their own respective organizations. Additionally, through the DevOps Federal Interagency Council (DFIC), IRS will continue to build relationships across the federal IT community, share best practices and lessons learned, and save time and resources by avoiding mistakes made by others.</p>

Smithsonian Institution: National Museum of African American History & Culture

 	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release		X			
	Testing		X			
	Security	X				
	Infrastructure			X		
	Delivery methodology		X			
	Culture and Governance			X		
Problem Statement and Goals	<p>The concept of DevOps was introduced to NMAAHC leadership team as a methodology which was complimentary to the agile processes the Web Development team was already following.</p> <p>Senior managers approved of and supported the methodology as they understood it to be an industry trend and best practice to increase the team’s productivity, optimize the project workflow, develop standardized practices and policies, introduce testing within the pipeline and incorporate time for documentation.</p> <p>NMAAHC management team conceptualized DevOps as a set of tools and best practices to progress from start-up mode to a sustainable model with respect to the delivery of the unit’s web-based products.</p>					
Approach	<p><i>How did your agency get started?</i></p> <ul style="list-style-type: none"> Identified business problem and existing process Evaluated the unit’s development workflow Reported on areas that could be improved upon by leveraging this methodology <p><i>What steps did you take next?</i></p> <ul style="list-style-type: none"> Technology research to learn what industry trends where addressing this business problem Captured suggestions and recommendations from peers and industry colleagues Continued conversation with NMAAHC IT leadership Created DevOps team which included me as a DevOps Lead and the appropriate staff members to server as System Admin & Security Manager Communicated changes and set expectations with web dev team to include the PMs, QA & product owner Engaged in conversations with SI Enterprise IT [OCIO] re: services and supports we could expect from the group as well as capture buy-in to allow our unit to transition our existing way of working with them to the new strategy. 					


<p>Benefits</p>	<p>Initially, the transition to DevOps was not intentional. Our organization’s IT web development group was already following the Agile approach for development projects but not efficiently or effectively. Additionally, the department had access to an underutilized AWS account.</p> <p>The gap analysis, feedback from our leadership team, my own experience and available resources suggested that there was an opportunity to add automation to our workflow, discover and leverage AWS resources to realize process improvements.</p> <p>The research for tools which included AWS identified the DevOps methodology as an approach we should look more into. Because we had the flexibility and no truly critical systems would break, a decision was made to ‘try it’.</p> <p><i>What benefits did you hope for when you started DevOps?</i></p> <ul style="list-style-type: none"> • standardize processes and procedures • have a more transparent communication process and more frequent check-ins between all stakeholders • create documentation to capture the underlying strategy and decision-making process <p><i>What benefits have you gained to date?</i></p> <p>The most valuable benefits have been ‘knowledge-gained’ of the methodology. This is has led to an understanding of the various components which feed into it the ability to have an intelligent and coherent conversation with enterprise IT about our unit’s needs and the services required for a smoother transition.</p> <p>The conversation with OCIO has set expectations that requirements will change over time and our unit is willing to share our lessons learned and assist with their adoption of a similar process.</p> <p>We have also widened our network of practitioners and forged partnerships with vendors and agencies in order to receive suggestions and recommendations along with the opportunity to hear pain points, ‘bad practices’ to avoid and simple sanity check of our ideas.</p>
<p>Lessons Learned</p>	<p><i>What would you do differently if you could go back in time?</i></p> <p>I would do a more comprehensive inquiry which would have included capturing feedback from other agencies and peers who had implemented or were considering this methodology.</p> <p>This would have highlighted the many moving pieces needed to be considered e.g. stakeholders who would need to input into the adoption and acceptance.</p> <p>This knowledge should also include the extensive research I had performed to educate myself of the concept and the process and policy changes that would impact current workflows and ‘ways of working’.</p>
<p>What’s next?</p>	<p><i>What next steps are you planning to take to advance your DevOps program?</i></p> <ul style="list-style-type: none"> • Complete initial draft of system and network architecture • Implement DevOps workflow for a less critical application as a proof of concept • Demonstrate/discuss with stakeholders how DevOps strategy integrates with current • Determine the use cases in which DevOps would not prove useful or introduce unnecessary overhead. • Work with security manager to develop and integrate Dev[Sec]Ops strategy.

National Park Service

	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release			x		
	Testing	x				
	Security			x		
	Infrastructure			x		
	Delivery methodology		x			
	Culture and Governance			x		
Problem Statement and Goals	<p>We didn't originally think we were initiating DevOps. Instead, we saw a duplication of effort. Our goal was to reduce overlap and address the disconnect between Dev and Ops.</p> <p>We've always approached this from a data management perspective with a goal to create tools that parks could use. This started with a focus on managing GIS data and other digital assets- including a library system. There were parallel dev projects going on in each of these programs. There were also very similar efforts going on at multiple parks. We knew there was no way we were ever going to get the resources to support each one individually. We had to think of more cost-effective ways to operate. DevOps has helped us think about how to operate at a more enterprise-level.</p>					
Approach	<p>We started by having conversations with other division/program chiefs. We needed to connect with them to initiate a more collaborative approach. Specifically, we had developers in two divisions. We created a blended "dev" team and hosted meetings and presentations for staff to get to know each other.</p> <p>Because we were responsible for the data center, we started seeing outside applications that people just wanted hosted. Some of them we already had solutions for. We didn't necessarily stop them once an application was built but seeing this duplication early on helped us realize the magnitude of the issue. We knew if people came to us earlier in the process with their requirements, we could help them avoid the development costs and point them to existing solutions.</p> <p>Historically, our dev side has been completely decentralized. On the ops side, users (parks or offices) would come to us for server space- not expecting or wanting input. They had become accustomed to easy, higher-level or privileged access. We needed to gain more control and take back some of the management and oversight for both security and compliance reasons and to reign in costs. Our SQL servers with GIS were one example. Before all parks would get their own virtual machine with the database and ESRI software and would have to manage it themselves. To change this, we needed to think through how we engaged with customers. We since consolidated the data centers into a central location to be more effectively managed.</p> <p>Since we started DevOps and created our blended teams, we're not doing manual deployments anymore. We have automated build processes. We still face a challenge around some of the public servers. We're still very early on in automating testing.</p>					

<p>Lessons Learned</p>	<p>We took an informal approach without a separate budget request. We didn't ask permission in a formal way. Taking a less formal approach helped avoid some of the bureaucracy at the beginning. A more formal approach would have required that we demonstrate the value or savings. When we started, we were in a knowledge vacuum. We would have needed to know the scope of the challenge, what it would take, or what the result would be. So, for us, it was better to start as an agreement between colleagues to work better together. Then, we were able to see the benefits and then share those as a viable solution. As we share our successes and savings, we're gaining more visibility. One thing that's been key to our success on the Dev side is truly understanding our architecture. From there, we were able to better set both Dev and Ops standards and reduce customization and rework. For example, we asked, "Will Ops give everyone their own SQL server or are we going to manage that? How do we handle the managed services we provide?" That's how DevOps comes together.</p> <p>Also, we asked, "To what level is Ops about providing the platform as a service? If the application that the Dev team is building becomes a service, as well? We needed a structured approach." And now, NPS is moving towards that.</p> <p>Doing this informally had another benefit of building support from the teams up. Sometimes when initiatives come down from upper management, there is a reluctance to buy-in because it seems like another thing to do. Instead, our teams went to the new mode of operations because they saw the direct benefit to them.</p>
<p>What's next?</p>	<p>We need to develop more process documentation. We need to focus more on defining and refining processes over tools. It's better to set the process and then find tools that support it.</p> <p>We also need our organization structure to reflect how we're working. For example, we have Dev and Ops teams but we have additional teams for security, project management, data, etc. In the future, we'd like to have integrated teams with a project management referee to help coordinate and align them. This would help us better deliver according to the customers deadlines.</p> <p>We want to define our standard products and services so we can communicate these to our customers.</p>

National Science Foundation

	Current State of DevOps Journey					
	Initiative	Initial	Defined	Managed	Optimized	Continuous
	Build, Deploy, & Release			x		
	Testing			x		
	Security		x	x		
	Infrastructure				x	x
	Delivery methodology			x		
	Culture and Governance		x			
Problem Statement and Goals	<p>We actually didn't start with the intent to launch DevOps. Instead, we started by moving from waterfall to agile. As we saw the value of moving to incremental deployments over having year-long cycles, we realized we wanted to move faster. Test automation was one of the first things we did. Knowing DevOps was a best practice, we wanted to bring in the infrastructure teams. It wasn't until we started expanding to more applications that we decided we needed a dedicated DevOps team. Prior to this, they were called the "tools" team. We then expanded to security. It wasn't until they started building containers and scanning things on every build until the security got fully involved.</p>					
Approach	<p>NSF is an incredibly collaborative culture. Every function and system they added was thoroughly socialized within their Division. The rest of the organization just saw the benefits of faster builds and deployments. Each incremental advancement was bought into by the management and the staff.</p> <p>The degree of buy-in wasn't universal across the organization, however. Some teams moved out on adoption faster than others. Today, we see the impact of having some inconsistencies in use and program maturity across the team.</p>					
Benefits	<p>The biggest benefit has been time-savings. This equates to more features being developed. Like others, we have a "never-ending" backlog. Before DevOps, developers were staying up until 2am on a Friday night to run validation scripts in off hours. Now, we have one engineer to click a button during the day and it runs. It's optimized the time of our most valuable resources.</p> <p>The morale of teams able to do "no downtime" deployments has gone way up. Late nights and weekends have been reduced and they're much happier at work. It's more of an intangible benefit but make this a much better, more productive place to work. Further, faster releases encourage better overall code writing because the feedback is almost immediate.</p> <p>Another thing that improves our team's confidence is having data on increased code coverage. They feel better about putting something out quickly that used to take three months in the old process.</p>					

<p>Lessons Learned</p>	<p>Our strategy to starting with one or two teams was good for building early wins and some momentum. One of the challenges of this approach was that it created variations in program adoption and maturity. For example, the processes got refined through lessons learned as it spread to other teams- leaving some applications behind. If we had more eyes on this in the beginning, we could have rolled out a more comprehensive solution in the first iteration.</p> <p>Further, the order in which was put in some of tools could have been more strategic. If we'd done more planning about the end state, we might have put in some of the front-end pieces differently. For example, if we'd all used Git instead of Subversion, we'd have been better off. In another example, we're getting our security scans in with the container builds now. Ideally, we would have prioritized that with the security tools- earlier in the process.</p>
<p>What's next?</p>	<p>Our next thing is going from Subversion to Git. We want to get all applications and teams to start having automated deployments and using the tools available to them.</p> <p>We're going to deploy our first container in the cloud in August 2019. It'll be exciting to have autoscaling and moving off-premise in AWS. We expect to see great optimization from this move. We struggled with cloud adoption a couple of years ago but were able to get past those barriers. Establishing a containers platform wasn't as difficult but it's been a long process.</p> <p>They're just now establishing a roadmap of how they're going to move everything to the cloud. It took nearly one year to create our "landing zone" approved at all the right places to start their majority roll out to go beyond the couple of applications they put in the cloud</p>

SHARED CHALLENGES AND TACTICS

DevOps is both a new way of thinking and a new way of working. It requires a growth mindset as much as it does new and reengineered practices and tools. In fact, agencies often hit barriers built by the old “that is not how we do it here” thinking. Without a motivated and optimistic set of leaders, practitioners, and supporters who believe change is possible, DevOps will not get off the ground.

Change agents who make the most progress have an open, collaborative approach, and empathy for those less certain of the possibilities and benefits. Open mindedness and a willingness to push through difficulties is the normal “best practice” among these common challenges with procurement, culture change, and communications.

Procurement

Agencies face procurement challenges because adopting a DevOps mindset means embracing a greater degree of ambiguity and uncertainty. The core issue revolves around the challenge of defining requirements, when specific requirements are not well known (intentionally), and fairly evaluating contracts.

Shared Challenges

- In the traditional procurement model, clearly identifying and documenting requirements upfront is critical. When utilizing a DevOps approach, agencies seek ideas and potential solutions earlier in the development process when specific requirements are not yet known.
- There is a legitimate concern within procurement and programs about ensuring Federal Acquisition Regulation (FAR) compliance.
- Program managers fear making costly mistakes (in time, money, functionality, service, etc.) by selecting a vendor without the needed skills, capabilities, and commitment.

Tactics

Shifting towards agile procurement techniques is highly recommended as it allows for greater flexibility around requirements. Agile procurement is a way of purchasing solutions according to the outcomes sought, rather than a product specification. It concentrates on allowing flexibility by focusing on explaining your problem and allowing contractors or providers to provide a vision for how to move forward. To execute effectively, it is necessary to avoid focusing evaluation on what will be created that will likely change over time, instead focus on how contractors plan to approach the current environment and challenges. The following are common tactics being leveraged to facilitate agile procurement in the federal space:

- Use statement of objectives (SOO) instead of statement of work (SOW) or Performance Work Statement (PWS) to provide open framework for contractors to deliver their approach.

- Invite contractors to a whiteboarding session between government and vendors early in the procurement. Bring the SOO and work together to remove assumptions and gain visibility from industry on how to approach the problem.
- Down select a group of contractors and then work with them to write the SOW.
- Issue coding challenges to have vendors demonstrate their problems solving approach and technical skills. In this situation, the agency will present a problem statement and ask vendors to respond with their best approach. Some agencies submit the problem and give vendors a certain amount of time to solve the problem and submit answers back to the agency, but some have chosen to make this a live session on site where they can assess the vendors in a simulated environment similar to where they will operate. While this has been deemed a very effective way to measure skillsets and delivery, one of the main criticisms around live coding challenges is that the amount of effort and investment required on behalf of vendors to participate can significantly limit the field.
- Increase the use of oral presentations to evaluate contractor approach and knowledge.
- Decrease the contract duration and value to minimize impact of a potential mistake.

Culture Change

The buzz around DevOps is often about tools. Automation tools promise to up speed, ease, and security for cumbersome processes. The innovation is exciting and energizing—particularly for the developers, system admin, and project managers diligently working through each manual step. But, of course, DevOps isn't just about tools. The collaborative spirit of DevOps and underlying agile philosophy doesn't come with an impressive demo but it's just as important. It's the culture change we make to how agency employees work together.

Shared Challenges

- Change can be difficult. It is common to avoid change initiated by others
- DevOps and the underlying processes and technology are complex. Colleagues and leaders may indicate they are onboard before truly understanding the proposed change or impact. People change their mind when reality hits about what exactly will be different.
- The current working environment (i.e., culture, performance management, business processes, etc.) at many agencies does not incentivize people to try new things or take risks necessary for innovation.
- Many people have a perception of limited resources. Time and funding dedicated to one effort may mean less is available for their project.

Tactics

Agencies should use techniques to shift mindsets such as:

- Lead change through shifting mindsets at executive and middle manager levels. If you can shift executive mindset, they are more likely to support, sponsor, champion, and enable the changes you are trying to make. If you can win over middle managers, they can act as crucial translator between executives and individual contributors to drive change and organizational mindset shifts.
- Determine how best to reach the individual and convince them of the benefits via “what is in it for me” (i.e., how the organization will save time and money, balancing priorities, driving change, achieving the mission, doing high value work etc.)
- Help executives and leaders understand the impact on people/individuals. Release window example when project teams have to be involved over a weekend. The stress of being “on call” can take its toll on people. The flip side is sharing the metrics/outcomes that improve performance.
- Pay less attention to what people *say* and more attention to what they *do*. Actions and behavior often provide even better insights into the values and beliefs of the organization and its employees than what people are saying.
- Run tests simultaneously on the current way and the new way. You are not asking for a commitment to change, just a commitment to try.
- Understand the current explicit and implicit incentives. Explicit include all of the compliance and policy issues. On the implicit side, people don’t want to get blamed for something, create rework, or reassigned to a different project. People fearing blame will be less likely to try new things. Develop a strategy/plan about what can be accomplished within a short time window.

MINDSET SHIFTS

DevOps has an associated growth mindset, as much as it has a set of associated patterns and practices. Enable change agents trying to shift the mindset of the organization to a target mindset more aligned with DevOps patterns and practices:

- From controlling change to embracing change
- From command and control to empowerment
- From functional silos to value streams and systems thinking
- From low-cost provider to enabling business value
- From hoarding resources to sharing and experimenting

You usually will not have the resources to do everything you want, but you will have some resources to do some of what you want. Identify the low risk plus low investment activities with largest potential impact. If you do enough of the small things well, you’ll earn trust and the right to do more. Focus on expanding your sphere of influence.

Empower your teams by giving permission to fail and ensure they have the right skills, right visibility, and right authority, all in support of making the best decision for their specific circumstances:

- Right skills include expertise or timely access to expertise in all aspects of the environment. This can include hiring for expertise, bringing in experts on short engagements, or training your team to build expertise.
- Right visibility means that information from the entire environment is available to the entire team (e.g., operational monitors can be viewed by developer centric team members, CI build results can be viewed by operations and support).
- Right authority includes permissions to modify the environment; but with accountability. At USCIS, for example, teams have permission to develop their own toolsets if they do not wish to use the approved set, but they're measured on usability and adoption of those tools to ensure they are delivering value and improving productivity.
- Understand the need for Cloud and Automation first approach to implement and pilot DevOps, Cloud infrastructure is elastic, which means if the pilot doesn't provide the promising results, the servers where the apps are built can be easily turned off without incurring further expense, and if the pilot turns out promising, the infrastructure handling the application can be scaled easily.
- With Development and Operations as one entity, the leadership must be a key player in bringing down the two silos, and encouraging team to work as one.

Communications

The role of the DevOps teams is to help lay the foundation for their projects and agency to embrace the core principles and ways of working. Communications is needed to inform, educate, and encourage more collaborative, integrated behaviors.

Shared Challenges

- Everyone is busy. People are reluctant to take on another process
- The attitude: "We are not Silicon Valley. We are DC. What works for them does not work for us."
- DevOps does not get a lot of attention in mainstream media because it's very technical. Most of what comes out is related to cybersecurity— with negative, fear-based messages.
- Methodology that has a multitude of definitions. Terms can be confusing.
- When the communicator does not understand the concepts well, bad information, or confusion spreads.

- It is difficult to get broad agreement on what DevOps is and what will be delivered.

Tactics

- Must educate and change the mindset to focus on the benefits not just to IT (including staff required to use the new processes) but to the mission/business side.
- Focus on positive messages, understand what people are reading in the media and where they might be getting their perspective. There can be a bias that comes from media coverage around cybersecurity. Make stories from known companies relatable. Pull stories from DevOps conferences - this is a great source of information on case studies and what not to do. Conferences may include DevOps Days DC, Agile conferences, etc.
- Determine where to share the word and ensure everyone is hearing the same message. Ensure solid understanding of concepts.
- Build relationships with leaders to help them understand this is all new and the teams need space to make mistakes.
- Sell the pilot concept first- undersell the benefits. Try, test, and monitor the impacts before overpromising. It is impossible to know what you do not know.
- Ask permission and time to test and make mistakes.
- Bring an outside expert or consultant for advice.
- Must have clear definitions and expectation on what will be developed upfront.

CONCLUSION AND RECOMMENDATIONS

There are hundreds of possible first or next steps an agency can take to advance their DevOps journey. Here are six recommendations collected and summarized by government and industry leaders.

1. **Say “Yes” to failure.** Failure in IT is traditionally associated with negative occurrences (server failure, backup failure, launch failure, etc.). However, in DevOps, failure is viewed as “opportunity”. Each by taking calculated risks each “opportunity” provides learning and context which informs innovation, learning, and improvement.
2. **Start small and build momentum.** Find one application or product team open to trying something new. A gauge of their readiness is whether they can say, “We will try. We want to figure out how this can be done here.” Inherent in the ability to start is having supportive IT and business leadership.
3. **Tell everyone you are doing an experiment.** Understanding this is an attempt to learn what works and what does not helps people be more accepting of failures and restarts.
4. **Show your work.** Make all of the effort needed to plan, implement, and assess progress in a visible manner. Make it clear where your teams are saving time and where they’re spinning their wheels and provide context. This will support your business case and allow demonstration of progress.
5. **Invite executives to see your teams in action.** Hosting retrospectives and show and tell meetings often (ideally monthly) helps generate interest and ongoing engagement. People at the top typically enjoy rolling up their sleeves for a while and seeing work in progress.
6. **Celebrate wins big and small.** Make sure you are transparent about the challenges and pitfalls, but balance this with celebrations for even the smallest wins. Product teams, in particular, are more likely to feel valued when their progress is publicly recognized.
7. **Be patient and preserve.** Anticipating the long road ahead can be overwhelming. Instead, advance the program one sprint or even one day at a time. Patience and perseverance is ultimately what leads to DevOps success.

GLOSSARY

Definitions for the following common DevOps terms were gathered from Channel Futures⁷.

Agile. Used in the DevOps world to describe infrastructure, processes or tools that are adaptable and scalable. Being agile is a key focus of DevOps.

Continuous delivery. A software delivery process wherein updates are planned, implemented and released to end-users on a steady, constant basis. It's the opposite of waterfall delivery, in which updates are released at an irregular, static pace.

Continuous integration. A process that allows software changes to be tested and integrated into a code base on a continuous basis each time a change is made to code. Most DevOps teams view continuous integration as an improvement over the traditional process of waiting until a large number of code changes are written before testing and integrating them.

Immutable infrastructure. An application service or hosting environment that, once set up, cannot be changed. If a DevOps team wishes to change a configuration on immutable infrastructure, the entire component must be re-initialized. While this may seem inefficient, the advantage of immutable infrastructure is that it makes environments more robust and reliable because inadvertent changes are impossible to introduce.

Infrastructure-as-Code. An approach to infrastructure configuration that allows DevOps teams to use scripts in order to provision servers or hosting environments automatically. This saves them from having to set up infrastructure by hand, a time-consuming and mistake-prone process.

Microservices. A type of application architecture in which applications are broken into multiple small pieces. For example, a microservices-based Web server might have its storage, front-end and security layers each operating as a separate service.

Serverless computing. A type of service that provides access to computing resources on demand, without requiring users to configure or manage an entire server environment.

ACKNOWLEDGEMENT

The DevOps Working Group thanks those who contributed to the development of this primer and provided invaluable feedback as reviewers.

Authors and Affiliations

This paper was written by a consortium of government and industry. The organizational affiliations of the authors and contributors are included for information purposes only. The views expressed in this document do not necessarily represent the official views of the individuals and organizations that participated in its development.

Christopher Weaver	Internal Revenue Service (Government Chair)
Robin Camarote	Wheelhouse Group, LLC (Industry Chair)
Annette Mitchell	DevOps Federal Interagency Council (DFIC) Lead, Internal Revenue Service
Ravyn Manuel	Smithsonian National Museum of African American History and Culture
Raj Dolas	U.S. Patent and Trademark Office
David Hernandez	Excella Consulting
Dhaval Shrimankar	11 th Hour Service
Nancy Delanoche	ACT-IAC

REFERENCES

¹ Amazon Web Services. *What is DevOps?* <https://aws.amazon.com/devops/what-is-devops/>

² Agile. 2020. In Oxford Online Dictionary. Retrieved from <https://en.oxforddictionaries.com/definition/agile>

³ Roberts, James. March 14, 2019. Comparing Waterfall, Agile, & DevOps. <https://dzone.com/articles/what-is-the-difference-between-agile-and-devops-1>

⁴ Vaughns, T. May 14, 2018. *The Importance of Agile Release & Iteration Planning*. Stable Kernel Blog. <https://stablekernel.com/the-importance-of-agile-release-iteration-planning/>

⁵ The graphic was created and published by Amazon Web Services (AWS). The original post can be found here. <https://aws.amazon.com/devops/what-is-devops/>

⁶ American Council for Technology-Industry Advisory Council. October 2016. *Legacy System Modernization: Addressing Challenges on the Path to Success* https://www.actiac.org/system/files/Legacy%20System%20Modernization%20Report%20v1.1%2001112017_0.pdf

⁷ Tozzi, C. April 5, 2017. Channel Futures. DevOps Dictionary: A Guide to DevOps Words and Terms. <https://www.channelfutures.com/technologies/devops-dictionary-a-guide-to-devops-words-and-terms>

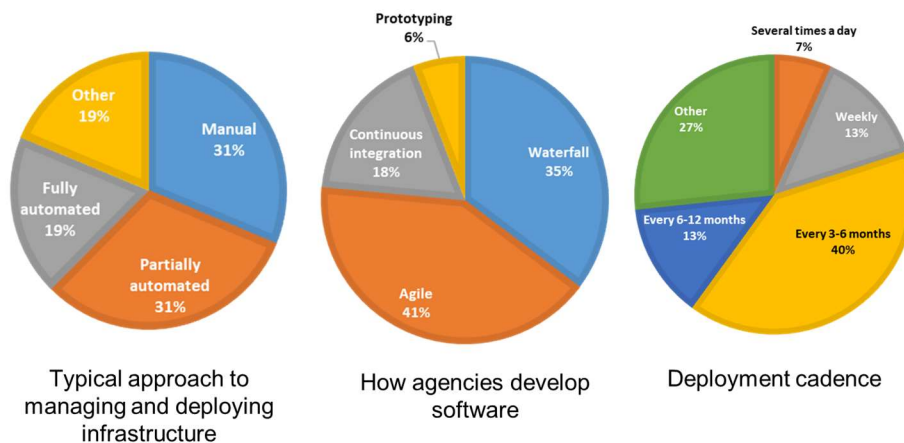
APPENDIX

ACT-IAC Federal DevOps Survey Results

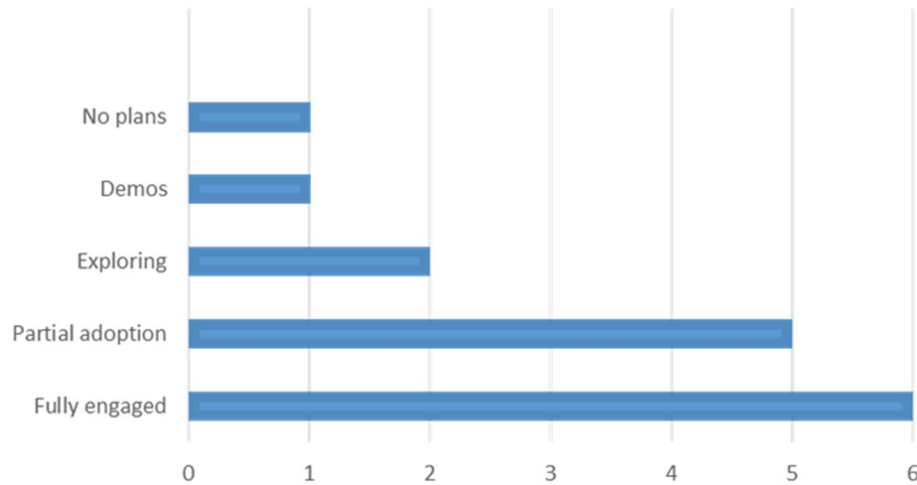
In 2018, the project team completed a survey on Federal experiences to date with DevOps. The results are included below. 27 individuals responded including 13 from government, 13 from industry, and 1 non-profit.

WHEN, WHERE & WHO?	<p>Emerging Technologies COI Chartered the DevOps Working Group to develop and distribute a short survey. Working with members of the working group a short (10-12 minute) survey was built, review and published to the ACT-IAC membership (and invited participants).</p>
PURPOSE OF THE SURVEY	<p>The DevOps Working group wanted to hear about agency's experience with DevOps, regardless of where they were in the journey. We sought to bring back "normalized" data to better ensure consistent reporting and better analysis.</p>

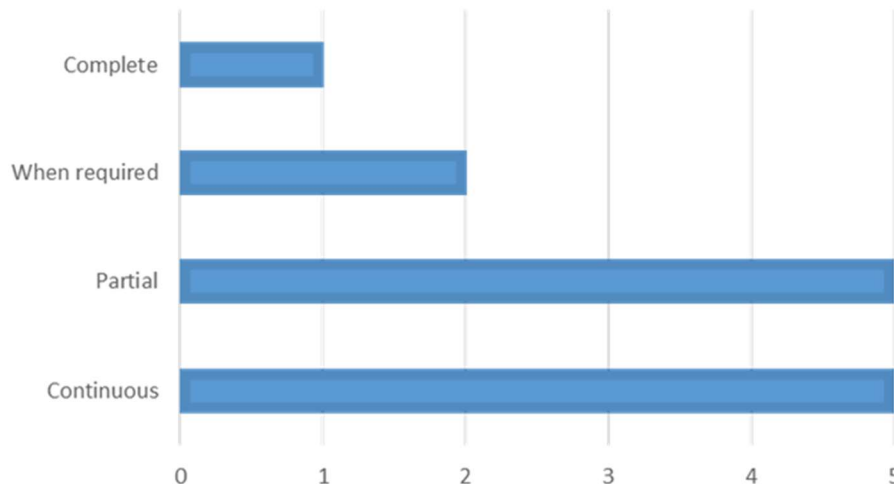
Responses on how products and projects were managing and deploying infrastructure, developing software, and deployment frequency are below.



WHERE ARE YOU IN THE PROCESS?



COLLABORATION BETWEEN TEAMS



Who develops and releases applications?

- Develops: 6 both contractors & government; 6 contractor; 1 other
- Releases: 9 both contractors & government; 3 contractor; 1 other

Has governance changed?

- 4 said yes
- 3 said no
- 6 said, “No, but it needs to!”

How has DevOps changed governance?

- 3 increased reporting;
- 2 increased attention to risks;
- 2 increased attention to security;
- 1 increased attention to procurement approvals

DevOps impact to internal processes (in descending order)

1. Automation
2. Security processes, improved security
3. Software delivery approach
4. Information sharing
5. Software delivery approach/security processes/customer experience