Micropurchase Deliverables for

DevOps Best Practices Task Order #59

On behalf of:

The U.S. Department of Veterans Affairs

Submitted by:

580 STRATEGIES
GOVERNMENT. TECHNOLOGY. SOLUTIONS.

March 11, 2020

TABLE OF CONTENTS

580 Strategies LLC
1100 7th Street NE
Unit #3
Washington, DC 20002
(415) 915-4679

U.S. Department of Veterans Affairs
810 Vermont Ave NW
Washington, DC 20571

March 11, 2020

Dear Mssrs. Croall and Ackerman,

Thank you for the opportunity for our team to advise and partner with you on defining DevOps Best Practices for the Department. We are excited about the innovative approaches you are taking to software development and hope that by lending our experience and expertise from commercial and nonprofit sectors to the body of knowledge the Department is building, we'll all raise the bar for services and product delivered to our veterans.

We have surveyed the landscape and used our own firsthand experience with many of these processes and tools to arrive at the following recommendations for adopting best practices to establish or improve upon the DevOps ecosystem at VA. We hope that you enjoy exploring these tools and experimenting with the myriad of ways they can help accelerate the delivery of software across your products.

Should you need additional help or expertise to develop any of these practices further, 580 Strategies stands ready to assist. Thank you for all you do on a daily basis to serve our Americas veterans--it's our pleasure to serve you in support of that mission.

Very Respectfully,

Rusty D. Pickens
Founder and Principal

Background and Task:

Responding to Microconsulting Solicitation #59, 580 Strategies is tasked with assisting the Department of Veterans Affairs in understanding best practices and tool selection for a modern software development and operations (DevOps) environment.

Background

As the Department of Veterans Affairs (VA) Office of Information Technology (OIT) transitions into Development Operations (DevOps), it is imperative that VA and OIT understand the best practices from both the public and private sectors to ensure it is implementing a modern DevOps environment. VA and OIT need to determine what tools best serve the VA's specific needs, whether they are provided by the government, contractor or a combination of both.

The Play

The play, or problem statement, for the task order is as follows:

> *"The Product Line Manager is seeking a modern DevOps environment. We need to understand best practices from the private and public sector, specifically around tool selection and integration methods, automation, and identifying the tradeoffs between government provided tools, contractor provided tools, and a hybrid approach."*

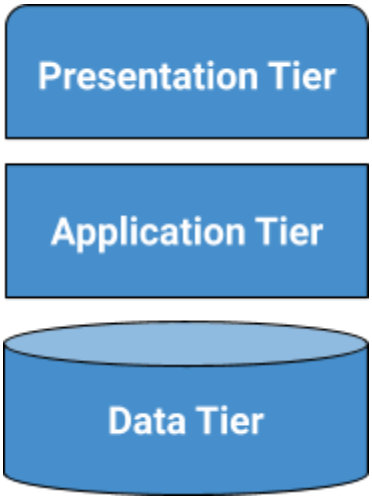
Our Approach

The Department already has a defined 10-step DevOps process that is working well for many teams. Given the overlap in tool sets at various points during the process, we approach the problem by examining each tier of a typical DevOps environment instead. For each tier, we discuss best practice approaches, outline the characteristics of proper tool sets for the functions needed, and make a recommendation for a tool to consider for use. In this way, the Department can easily adopt the recommended tool in order to get started quickly, but we've also provided the characteristics of useful tool sets so that as technology changes over time, this deliverable continues to provide long lasting value.

DevOps Tools and Best Practices:

To support the Department of Veterans Affairs growing DevOps ecosystem, the selection of a standard set of practices and tools to draw upon is critical to accommodate flexibility for teams and projects as efforts grow and wane over time. We will discuss each tier of the technology stack, offer options for currently available tools, and suggest our recommended tool for each purpose.

As modern technology applications are generally developed in a traditional three-tier environment, we examine best practices and tools through this lens. Keep in mind that most DevOps ecosystems have many copies of this type of arrangement: horizontally scaled copies for parallel development across multiple developers or teams and vertical copies for testing, integration, and release management purposes. However, the practices and tools for each tier remain largely the same no matter how many times they are replicated for scale.



The most effective approach to tool selection is to work from the bottom up as database selection and tooling is the backbone of most mature software development projects.

Database Tool Selection:

Most modern database engines offer a paired visual design tool. When the option to use a tool from the primary provider (such as MySQL, Microsoft SQL, or Postgres) is available, that tool should generally be the first choice. Tool sets offered directly from service providers generally have support for all features and are released in tandem with service updates to keep things current.

There are two differing approaches to contemporary database systems: NoSQL and relational databases. NoSQL databases are document-based and best suited for applications that contain large quantities of unstructured data. Relational databases have extensive query capabilities and are optimal for analytics and data science engineering. For most applications,, we recommend a relational database engine unless there is a compelling project need for a NoSQL approach.

Our Recommendations:

- ***MySQL Workbench*** for MySQL implementations
- ***SQL Server Management Studio*** for Microsoft SQL Server implementations

Integrated Development Environment:

Once the database layer is chosen, you can choose an Integrated Development Environment (IDE) to best suit your application/code layer for programming. Many modern IDE's also integrate database exploration and support functions directly into the tools in order to streamline development. These database tools do not need to be universal across the business because they do not result in work that is persisted to a project. Developer familiarity and ease of use are the properties to look for in a good database tool set.

When it comes to IDE's, the best feature sets tend to come from widely available commercial products which may or may not come from the primary language provider. For proprietary languages like C# (prior to .NET Core) or Apple's Swift it is best to use the IDE provided by the language developers whenever possible. These native tools tend to include support for all language features and will often be pre-released when language updates occur.

For all other languages like Java, PHP, Python, and Ruby, JavaScript and HTML, IntelliJ offers perhaps the best IDE environments on the market today. Their standardized IDE's make it easy to switch between projects or modules within a multi-language project and not need to worry about changing keyboard shortcuts or becoming accustomed to a new layout. While seemingly innocuous, these changes can greatly increase development time when switching environments due to developers having to adapt to the new tool set.

Our Recommended IDEs are:

- ***Microsoft Visual Studio*** for C# and .NET Core
- ***Xcode*** for Apple Swift
- ***JetBrains*** for other languages (Java, JavaScript, Ruby, Python, PHP, etc.)

Once database and code IDE's have been chosen teams can look at tooling and platform support for infrastructure, deployment and development.

Code Storage and Source Control:

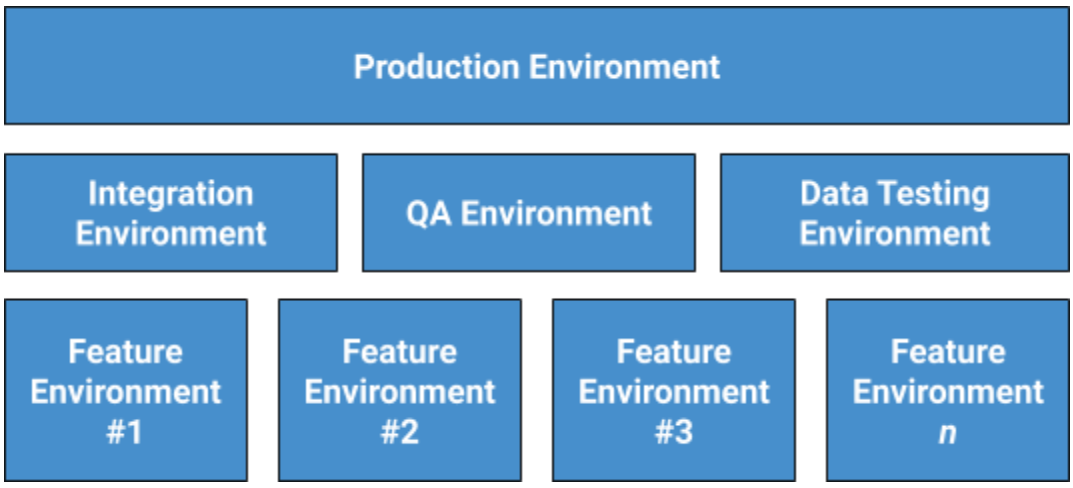
Any mature software development project demands code storage coupled with source control--and for the Department's processes this is the case as well. It's important not only to be able to have visibility into code history but also reasonably roll back changes when defects occur. Overwhelmingly, the industry is moving to Git as the source control methodology of choice as it has a large, rich, featureset and is easy to get started using. There are many cloud hosted Git providers available, but the market leader is without a doubt GitHub. It is quickly becoming the industry standard for code storage and management and given the Department's already positive track record with this tool, we see no reason to consider other options for this purpose.

Git and GitHub empower multiple aspects of the Department's DevOps process including allowing developers to experiment within their own feature branches, submit new code for review through Pull Requests, and it combines nicely with Continuous Integration and Continuous Deployment tools for streamlined releases.

Our Recommendation: ***GitHub***

Infrastructure:

The first stage of infrastructure analysis generally looks at the servers and/or hardware that runs the system workload. Historically, developers ran full environments on their local machine which often entailed hosting databases and application servers locally. This generally meant that runtime environments were based on local hardware specs unique to the developer's system and code may behave differently on that system versus in production. Modern DevOps techniques are moving away from this approach to an image-based and much more standardized process that minimizes configuration differences between development and production environments.



In translating the traditional bare-metal server approach that had specific hardware environment tightly coupled with each tier/function of the DevOps environment stack, we now see developers and

engineers adopting lightweight “containerization” of lower environments and infrastructure-as-code for middle-tier and production workload environments.

Containerization and infrastructure as code provide a standard and structured means of quickly provisioning new environments for development, testing, and deployment in a manner that gives teams high confidence the system environment itself is configured the same way--and will behave the same way--every time.

Containerization with Docker:

With the move to the cloud and the need to better describe servers and environments, containerization has become a critical part of almost every modern tech environment. The leader in the space is unquestionably Docker. Their technology allows for servers to be described, built, and exchanged using a common language, framework, and source control (via DockerHub). Docker images allow developers to locally run their systems with configurations identical to those used in production which helps to prevent runtime issues. Docker images can generally be built for any server environment so long as the environment is clearly defined.

A good Docker image *should*:	A good Docker image *should not*:
• Be portable • Be reusable • Be minimal and lightweight • Stable and free from bootup issues • Configurable environments	• Require user interaction to boot up beyond a “run” command • Contain unneeded packages or libraries • Require any workarounds to function in a developer environment

Once the Docker image has been built, DockerHub can be used to exchange and deploy the image as needed. Docker Hub acts much like Git for pre-built images where tags and similar functionality can be used to control deployment and releases.

Our Recommendation: ***Docker***

Infrastructure as Code:

While Docker is our choice for server containerization, modern cloud environments running higher tiers of DevOps environments often contain other resources that leverage modern cloud features. These resources may be for storage, extra computational ability, multi-zone databases or even just load balancers and firewall rules. Traditionally, these were managed manually and often provisioned in accordance with a large specification document. The classic process, while productive, left wide latitude for error. In some cases resources get spun up and never added to the documentation or deprecated resources are never removed from the environment. Downstream, this causes issues when building development environments and troubleshooting environment related issues.

The industry is moving to the concept of “infrastructure-as-code” which allows engineers to describe an environment programmatically and full copies of that environment can be quickly spun up or down as

needed with high confidence they are identical each time. This allows rapid, production-scale environment testing and quick resource allocation for testing and even local debugging. It is our advice that when migrating platforms to the cloud an infrastructure-as-code solution be strongly considered.

Our Recommendation: ***Terraform***

Continuous Integration and Continuous Deployment (CI/CD):

A well architectured and stable DevOps environment should have some method for continuous integration (how new or updated code is merged into a single, contiguous codebase) and continuous deployment (how code is moved between environments).

There are many providers in the marketplace who deliver similar products. Most of the major providers are platform agnostic so long as you can provide a stable container environment to run automated testing against. Without stable and properly architected containerization, it is difficult to properly execute CI/CD. Some providers sell hardware access combined within a CI/CD service, while others separate the machine instances from the service layers. Services that only sell an application to control CI/CD almost always have more favorable pricing models.

A good CI/CD platform *should*:	A good CI/CD platform *should not*:
• Accept containers in various formats • Allow for easy configuration and runtimes • Connect to all desirable source control providers • Provide good logging and reporting of CI run events • Well integrated with version control systems • Allow for any kind of deployment scripts to be run against any major cloud provider • Be configurable per-branch, release tag, or other similar triggering event • Have a scalable pricing model	• Only allow running a single job at a time • Suppress logs and events that might be relevant • Require manual uploading / connection to environments • Allow partial deployments

Our Recommendation: ***Jenkins***

Platform Monitoring:

In today's marketplace there are more monitoring tool options than any other single tool type. Many monitoring tools provide a verbose set of analytics but can easily be misused or poorly configured which often leads to purchasing more tools than needed. While each team's individual requirements may vary, a good reporting platform should provide insights into database performance as well as server/environment performance with minimal setup.

A good environment monitor should:

- Provide query level performance monitoring on the database
- Provide CPU and Memory Pressure monitoring on all hardware/VM instances
- Monitor network speed and response type of API or other web servers
- Provide the ability to create custom dashboards to monitor real time metrics across the entire application landscape

Our Recommendation: ***DataDog***

Log Capture:

In order to properly troubleshoot issues in production, it's important that logs are captured, searchable, and auditable. The industry standard approach is to have an ELK (ElasticSlack, LogStash, Kibana) stack to ingest logs from applications. This allows engineers to quickly and effectively find and fix bugs when production issues occur. Many of the log capture tools on the market offer similar feature sets and most if not all can be interacted with no matter the choice of tech stack.

A good log monitor should:

- Allow free text searching
- Allow query style roll up and searching
- Provide at least basic log analytics
- Allow log based alerts to be configured for key logging events
- Be capable to capturing logs from most major tech stacks or logging libraries

Our Recommendation: ***SumoLogic***

Summary and Adoption:

Combining these best practices and recommended tools at the appropriate tier of the ecosystem and at the necessary points in the Department's DevOps process can yield powerful results to get working software into the hands of veterans faster as Veterans Affairs continues to embrace cloud computing and DevOps.

While this deliverable makes recommendations at every level of the DevOps ecosystem, it is entirely possible to adopt a *Crawl* > *Walk* > *Run* approach to implementing them one-at-a-time or as opportunities present themselves. We look forward to continuing to help the Department explore, expand, and enhance the use of these industry leading approaches and tools to provide faster and higher quality service to our community of American veterans.

PROJECT TEAM

Rusty D. Pickens - Founder and Principal

Rusty D. Pickens is the Founder and Principal of 580 Strategies. He is the former Senior Advisor for Digital Platforms at the U.S. Department of State, and former Acting Director for New Media Technologies at the White House, where he led teams who operated cloud platforms for the Obama Administration to increase public engagement, improve user experience, enhance staff productivity, and heighten security posture. During this time, Rusty created new systems for and built new teams to lead Whitehouse.gov, the White House email outreach services, the Presidential correspondence system, the We The People petitions system, the White House Appointment Center, and the U.S. Embassy contact management systems.

Rusty's two decades of leadership experience aligning organizational vision with technology strategy across top federal agencies and start-up environments included the Federal Salesforce Community of Excellence, the U.S. Small Business Administration, the 2009 Presidential Inaugural Committee, Obama for America 2008, and the Chickasaw Nation of Oklahoma. He currently advises clients on unlocking the potential of cloud computing and agile software delivery to vastly improve their digital presence and citizen experience.

OVERVIEW / 20+ years of leadership experience in aligning organizational priorities & vision with technology strategy across top federal agencies and start-up environments impacting 20K+ stakeholders. Advise on and executed innovative solutions nationwide with net worth of $80M+ while exercising fiscal responsibility in procurement and budget strategies. Engage with high-level stakeholders and top corporate executives across industries during career span.

EXPERTISE / Executive Digital Strategy and Operations | Organizational Development | Budget/Contract Management | Agile Software Delivery atop Cloud Platforms | Infrastructure Architect | Large Scale & Rapid Deployments

---

David Colangelo - Software Engineer

OVERVIEW / Dave is a solutions-oriented engineer with experience designing systems, applications, websites, databases. He is currently the Managing Partner of P&D Technology Solutions, which he started to bring better software to everyone. After spending time as a developer and project manager at various large companies in the commercial private

sector, he decided it was time to strike out on his own and bring his industry knowledge to everyone.

Dave has had the opportunity to work on large scale data processing and storage projects for the environmental industry. Payment and PCI compliance applications for the credit card industry as well as high-scale code scanning and security solutions for software packages. AWS is Dave's platform of choice and he has architected numerous solutions that now run at scale in various high capacity environments.

EXPERTISE / Large scale Java API architecture and development, highly scalable database layers, elastic search optimization and integration, large scale data processing and big data analytics, end to end security and system architecture.

Peter Hicks - Software Engineer

OVERVIEW / Pete is an experienced software engineer with over 7 years in software engineering, cloud infrastructure, and developer operations. Familiar with many different software environments and industries. Proven track record in building, leading, and managing global teams. Confident in bringing new solutions to market and working with companies of all sizes. Comfortable in front of a development environment and providing technical leadership for a complex software ecosystem. Extensive experience in enterprise level and custom software solutions for private clients. Major contributor to startups that have had successful exits and greater marketplace success.

EXPERTISE / Rails, React, Redux, NoSQL/MongoDB, REST APIs, Node.js, Angular.js

---