HOW GOVERNMENT AGENCIES CAN USE CONTAINERS FOR IT (NOT JUST SHIPPING)

Sanjeev Pulapaka and Munish Satia

June 2018
# TABLE OF CONTENTS

Introduction ............................................................................................................................................. 1

What is Containerization and Is It More Efficient? ................................................................................. 2

Challenges and Best Practices ................................................................................................................. 3

- Challenge 1: Monolithic Architecture & Legacy Technology .......................................................... 3
- Best Practice for Resolution: Migrate to a Microservices Architecture ........................................ 3
- Challenge 2: Infrastructure Constraints ............................................................................................ 5
- Best Practice for Resolution: Setup Infrastructure Required for Container Operations ............. 6
- Challenge 3: Lack of Streamlined Processes ..................................................................................... 7
- Best Practice for Resolution: Change Dev and Ops Processes ....................................................... 8
- Challenge 4: Organization Culture and Knowledge ......................................................................... 9
- Best Practice for Resolution: Manage Organizational Impacts ..................................................... 9

Additional Considerations & Conclusion ............................................................................................... 10

References & Further Reading .............................................................................................................. 11

About the Authors .................................................................................................................................. 11
Introduction

The efficiency of government technology seems to lag behind the private sector. One key reason is the tendency for technologists (especially those serving the government) to operate in silos, with distinct differentiation between development, infrastructure and operations teams. The reason that this problem is more prominent in government is that Government agencies seek a separation of responsibilities to avoid risk. That same risk aversion causes agencies to use inefficient processes for longer periods than the private sector. The risk aversion to newer, leaner processes is driven by a fear that the such processes may compromise security and needs for traceability/accountability to the public.

The proposition behind this white paper is that this risk aversion, and the siloes that result can be resolved using a technological innovation referred to as a ‘container.’ Containers make it possible for agencies to invest in automated and efficient DevOps practices and processes, without sacrificing traceability, accountability or security. This paper offers an overview of containerization, discusses some of the challenges in adopting containerization within government agencies, and offers some best practices to help agencies adopt containerization and derive the maximum benefit from that adoption.
What is Containerization and Is It More Efficient?

**Containerization** is an approach of bundling applications as well as their dependencies, into a software package called a container image. This image is executed as a Container process (or more simply, a Container) running on a special software engine that interfaces between the container and the operating system. The container can be moved easily between environments and operating systems/platforms. **Figure 1** illustrates an executing container. Once the container image is created, it cannot be changed.

Hardware virtualization made it possible to place several different tasks on the same computer server by creating multiple “Virtual Machines” (VMs) on one physical server. Containers have the same consolidation benefits as that hardware virtualization, but with several distinct advantages – including potential significant efficiency advantages. First, containerization technology reduces the need for duplicate operating system (O/S) code. Second, spinning up / launching applications is much quicker, taking just a few seconds for containers, versus a few minutes for VMs. This is because a container does not need to instantiate a new operating system for each workload. Third, containers can be orders of magnitude more efficient in use of capacity, because they take up a very small footprint (typically just megabytes, versus gigabytes for a typical VM). This makes it possible to run multiple containers on a single machine, thereby making much more economical use of infrastructure. Fourth, containers are flexible, and can executed on either VMs or physical “bare metal” servers. Thus using containerization can provide significant resource savings and increased efficiency. The larger the footprint of the O/S, the greater the potential savings available with containerization. For example, containers could create savings of 25% over VMs on resources where the O/S footprint is 50% of the overall system.

Containerization is especially beneficial for increasing efficiency of development and operational processes in a multi-vendor environment.
environment through the increased portability as well as traceability offered by containerization. However, the technology introduces fresh challenges in the development, operations and management of IT at government organizations. The next section discusses these challenges as well as best practices though which these challenges can be overcome. It will provide a path to a dynamic, efficient and cost-effective IT organization.

**Challenges and Best Practices**

**Challenge 1: Monolithic Architecture & Legacy Technology**

The technology landscape of many government organizations includes a complex mix of legacy applications built on older frameworks such as Oracle Forms, .NET, and more modern applications using web application architecture. Most of the older technologies are not compatible with effective methods of containerization. For example, containers cannot be built for Oracle Forms and while containers can be built on .NET, the resulting container is very resource-intensive, nullifying the goal of optimizing resources and reducing costs. The same issue even extends to several of the relatively modern web based applications. The designs and architecture of some of these applications are monolithic in nature and do not allow for easy separation of the application into independent modules, which in turn limits the feasibility of containerization. For example, a modern system developed for managing workflows and tasks of three related business processes might have a single front end and a single package of application code, and might access a single database. When any one of those three processes needs to change, the entire package would need to be redeployed and extensive regression needs to be done to handle potential database and front-end changes. In this situation, containerization would not work well because the processes cannot be broken apart easily and placed in separate containers. In fact, in this example, containerization could increase production problems because of the limited flexibility and scalability of the application. The use of a single database structure also results in the creation of large and heavy databases that cannot be easily containerized.

**Best Practice for Resolution: Migrate to a Microservices Architecture**

Government organizations facing the above challenge should look at adopting a microservices architecture to migrate from the existing monolithic design & architecture to a modularized application architecture with modern technology. **Microservices architecture** is an approach to application development in which a large monolithic application is broken down into an ecosystem of simple, well-defined modules called microservices that are not dependent on each other. The conceptual architectural difference between the monolithic and microservices architecture is illustrated in Figure 2 below.
To accomplish the autonomy of a microservice, industry best practice is to use Domain Driven Design, an approach that helps break down complex business functionality into smaller components by defining boundaries for related business and technical topics. Called bounded contexts, these boundaries form the basis for development of microservices. The main advantage of this approach is that it builds tremendous flexibility for changing business logic quickly and independently without impacting other modules or applications. For example, a regulation change may result in the need to change Function 1 of the microservice application illustrated in Figure 1. In this case, the microservice containing Function 1 can be changed independently without requiring changes to the microservices for Function 2 and Function 3. An additional benefit is the ability to adopt a best of breed approach for selecting the technology required for each microservice independently. This makes it possible to upgrade technology over time as newer options are introduced, thereby reducing cost and risk of future enhancements, and minimizing disruption to business operations. For example, after the system has been successfully operating for 18 months, let’s say we learn about a new business rules engine that could improve administration of the complex rules making up Function 3. Containerization allows the easy upgrade of that microservice to use the new business rules engine, without requiring that we make any change to Function 1 or Function 2.

The mutually independent nature of the microservices architecture offers an excellent building block for effective containerization. Each microservice module can be easily packaged into a ‘light-weight’ container that can execute independently, as illustrated in Figure 3, with the ability to be started and stopped as an independent process.
How Government Agencies Can Use Containers for IT (Not Just Shipping)

The small footprint of the microservices reduces the processing power required to execute containers, significantly increasing the possibility to optimize and maximize infrastructure resource use. Each containerized microservice can be prioritized and implemented in an iterative fashion, through projects that are appropriately scoped to bring benefits quickly to the organization.

**Challenge 2: Infrastructure Constraints**

Implementation of a containerized ecosystem for a large program could result in a need for hundreds of thousands of containers. The sheer number and resulting complexity could increase the prospects of a security attack, either from outside the environment or within the environment itself. It also increases the amount of network traffic, potentially affecting performance. Troubleshooting applications and finding the cause of a problem amongst the vast number of container processes would be a challenge. For effective issue detection and resolution, the infrastructure should be able to handle complex operations such as shutting down a container while letting another container of the same image execute in parallel.

Another issue frequently encountered in government organizations is separation...
between production and test environments. Containerization aims to solve the problem of portability across environments. But, if the environments themselves are completely isolated from each other without the ability to access container repositories, the benefit will be limited.

**Best Practice for Resolution: Setup Infrastructure Required for Container Operations**

There are several tools in the marketplace that can assist government organizations in the management and operations of containers to address the infrastructure challenges identified above. These tools are shown in Table 1, on the next page.

When adopting containerization, the type of container and associated developer tools are amongst the first choices that need to be made. Currently, the most prominent technology in the marketplace is Docker, specifically the open source Docker Engine. Developers tools are needed to define the operating environment and steps for building their code into a Docker image in a Docker file. Developers also need tools to define the interdependencies of an application consisting of multiple containers as a ‘compose file.’

Amongst all the technology available, the use of orchestrators for production-ready applications that are composed of containerized microservices is critical. Orchestrators provide the ability to start, stop, spin up and shut down containers as needed to ensure effective management.

For example, if a process fails and the container ends, the orchestrator can create another container to replace the failed process. Deployment of images to an orchestrator requires a container registry. The registry is a central place where tested container images are stored and versioned, providing the ability to deploy the container in multiple environments without any changes. Other critical tools needed are container security tools for policy enforcement, vulnerability scanning, patching, automatic audits, and threat protection, as well as container monitoring tools that can monitor containers and provide metrics for troubleshooting.

Agencies can choose to select the tools listed in Table 1 individually (on-premise and/or in the cloud) in an “a-la-carte” fashion. Alternatively, an agency can choose to adopt them as a cloud-based group called a Containers-as-a-Service (CaaS) platform. CaaS platforms provide the entire set of containerization tools, including management, deployment, orchestration and security, as needed. Current CaaS platforms offerings include Amazon ECS, Red Hat OpenShift, Apcera and Google Cloud Engine. One of the major advantages of using CaaS is the elimination of time-consuming tasks needed to setup, build and test the container infrastructure. CaaS providers use tested and proven automated processes to provision their subscribers’ container environments. This allows the organization to begin rolling out containerized applications almost immediately. Selecting a CaaS platform also
How Government Agencies Can Use Containers for IT (Not Just Shipping)

<table>
<thead>
<tr>
<th>Technology</th>
<th>Benefit/Purpose</th>
<th>Potential Tools to Use</th>
</tr>
</thead>
<tbody>
<tr>
<td>Container</td>
<td>Application or service, its dependencies, and its configuration (abstracted as deployment manifest files) are packaged together as a container image.</td>
<td>Docker, Windows Server Containers, Hyper V containers</td>
</tr>
<tr>
<td>Orchestrator</td>
<td>A tool that simplifies management of clusters and container hosts.</td>
<td>Mesosphere DC/OS, OpenShift, Kubernetes, Docker Swarm, and Azure Service Fabric</td>
</tr>
<tr>
<td>Registry</td>
<td>A service that provides access to collections of container images</td>
<td>Docker Hub, Azure Container Registry, Docker Trusted Registry, Private Registry</td>
</tr>
<tr>
<td>Development Tools</td>
<td>Development tools for building, running, and testing containers.</td>
<td>Compose, Docker EE, Visual Studio</td>
</tr>
<tr>
<td>Container Security</td>
<td>Tools for policy enforcement, vulnerability scanning, patching, automatic audits, threat protection</td>
<td>TwistLock, AquaContainer, Stackrox, Sysdig</td>
</tr>
<tr>
<td>Container Monitoring</td>
<td>Manage and monitor hosts and containers; provide performance metrics for troubleshooting.</td>
<td>Marathon, Chronos, Applications Insights, Operations Management Suite</td>
</tr>
</tbody>
</table>

reduces risks associated with lock-in to an individual tool in the rapidly evolving container technology landscape. Agencies with security concerns around sensitive data can consider a hybrid cloud approach, using CaaS in their own data center.

**Challenge 3: Lack of Streamlined Processes**

In government organizations, Dev and Ops teams typically work in silos, in which Dev teams create the software in isolation and hand off (release) to the Ops team for deployment in a separate environment. The Dev team is focused on creating software that is flexible enough to adapt to changing business requirements, whereas the Ops team strives to make sure the environment is stable, reliable, continuously available, and unchanging. Although many Dev teams in government organizations are migrating to an agile approach, most Ops teams have yet to make this shift. Their processes are unequipped to support continuous container-based handoffs and deployments. Consequently, projects can suffer delays to resolve conflicts and quality issues, nullifying efficiency gains from containerization. This issue is compounded by the fact that governments often have separate environments from those used by their supporting vendor(s). Significant amounts of time and effort are spent making sure that the environments are synchronized, with cross-checks and validations further limiting the speed of deployment.
Best Practice for Resolution: Change Dev and Ops Processes

Containerization allows Dev teams to package all their code and necessary dependencies in one container and then automatically deploy that container across multiple environments (e.g., for development, quality control, and production), either on-premise or in the cloud. Consequently, the time spent installing and configuring dependencies is eliminated, dramatically reducing deployment time and increasing efficiency. The need for extensive cross-checks and validations across environments is eliminated. Instead, the agency’s IT operations team can choose to focus solely on infrastructure optimization and maintenance – instead of deployments and configuration. These changes require a deployment process that is quite different from the traditional deployment process. This new process is depicted in Figure 4, below.

The ability to automate as many steps as possible throughout the process becomes paramount in this scenario. In a containerized workflow, developers write code and create a container image in their local development environment, execute and test the container image locally to ensure that the application is working properly, and commit the source code as well as the container configuration file to a Git repository. It is essential that an automated build process be configured to create container images from the source code committed to the Git repository and test the code contained in the images by running automated unit tests. On successful execution of the automated tests, the container images should then be published to the agency’s global container registry, a central place where tested images are stored. The container image created at this stage can be moved and executed in the next environment as is, without any changes. This process saves both time and resources by eliminating the need for vendor handoff activities like configuration

Figure 4: DevOps Process with Containerization
management (CM) walkthrough, checklists, and database handoffs. More detail on a fully automated DevOps process is provided in a separate white paper titled “DevOps CI/CD in the Federal Sector.”

Challenge 4: Organization Culture and Knowledge

One of the biggest challenges in working with containers is a lack of staff familiarity with the technology. The container ecosystem is growing, with the number of tools and technologies available increasing, making it challenging for employees to stay up-to-date and educate themselves about what is possible in a container environment. Another challenge is the organizational culture and structure of teams. With containerization, the focus on software deployment shifts towards developers, while infrastructure teams become mainly responsible for setting up and maintaining environments. However, traditionally organized IT teams may be unable to make the transition required to the new roles. Questions about job security and organizational politics often compound this issue and lead to resistance in adopting the new process.

Best Practice for Resolution: Manage Organizational Impacts

Government agencies should understand that adopting microservices and containerization in an organization is a journey that requires cultural evolution, one that needs careful nurturing and support. Teams should be ready to think and operate differently, adopting an agile and continuous deployment mindset.

To facilitate the transition, agencies can consider the advantages of creating vertical teams for each microservice, where the team owns the API, business logic, and the data layers. This is illustrated below, in Figure 5, below.

![Figure 5: Microservices based team structure](image)

In this scenario, the API layer is used for establishing contacts between different microservices/teams, allowing the independent functioning of each team. Containerization furthers this cause by decoupling infrastructure needs from development. A small, centralized infrastructure team can manage the entire

---

In this scenario, the API layer is used for establishing contacts between different microservices/teams, allowing the independent functioning of each team. Containerization furthers this cause by decoupling infrastructure needs from development. A small, centralized infrastructure team can manage the entire
infrastructure with their main responsibility being planning, acquiring and optimizing infrastructure. Each microservices team would own responsibility for everything else from development to operations, including testing and product management. These teams typically include no more than 10 cross-functional, highly skilled people, occasionally referred to as “two-pizza” teams. (So named because the team is small enough that it can be entirely fed by two pizzas.) The nimble and independent nature of these teams can result in vastly accelerated delivery, in addition to providing the agile flexibility needed to adapt to changing business needs. This structure also indirectly ensures quality of code, because the development team is directly responsible for the quality of their service in production.

Another critical consideration for the government is to focus on providing training on the latest technologies for IT employees as well as IT managers. This doesn’t have to be an expensive proposition as there are plenty of free or low-cost trainings available online that can be leveraged. Doing so enables government staff to truly understand the issues, options, choices and tradeoffs during implementation.

Ultimately, government agencies should realize and recognize that the procedural nature of the operating environment is likely to throw up obstacles that must be overcome through strong management backing and inspirational leadership. Agency leadership should also ensure that organizational change management processes are undertaken well in advance of the initiative.

**Additional Considerations & Conclusion**

Prior to implementing containerization, agencies should first fully understand the objectives, benefits and results they seek from containerization and the changes it will bring to the organization. Agencies may benefit from an initial engagement that studies the existing environment, presents the various technology options to stakeholders, defines a strategy and implementation plan for the organization and implements a pilot initiative with the selected technology. This step will enable stakeholders to firmly understand requirements before initiating a full-scale change. It will also enable stakeholders to understand the organizational and cultural aspects of the implementation and the extent of change necessary.

Despite its relatively novelty, containerization is rapidly changing the technology landscape today and government organizations should examine the technology seriously sooner, rather than later, to take advantage of the benefits it offers. These benefits primarily include speed to deploy new applications, ease of upgrading one piece of an application without redesigning/replacing the whole application, and lower cost/risk associated with multiple computing environments. These benefits are particularly important to government agencies because laws and
policies tend to change one facet at a time, while government’s technology has tended to be much less flexible / more monolithic. Careful planning and investment aligned with the best practices discussed in this white paper will successfully position government organizations to leverage containerization in their mission to efficiently and effectively serve the American public.

References & Further Reading

“DevOps CI/CD in the Federal Sector”
“Cloud First... but Not Always”
Containerization Docker Application Lifecycle with Microsoft Platform tools

About the Authors

Sanjeev Pulapaka has extensive experience developing technology strategy, defining the architecture, and leading the design and implementation of high impact technical solutions for commercial as well as government organizations. Sanjeev holds an MBA from the University of Notre Dame and B.S from the Indian Institute of Technology, Roorkee.

Munish Satia has more than fourteen years of progressive experience in the architecture, design and development of enterprise scale applications and cutting edge next generation grants management solutions. Munish holds a Masters in Computer Applications from Maharishi Dayanand University and an MBA from Guru Jambeshwar University.

For more information or advice about Containerization, DevOps and Agile, and other technology topics please email info@reisystems.com, or call (703) 480-9100.