23rd ICCRTS

"Multi Domain C2"

Using Automation to speed up software delivery to the fleet

Topic 5: Highly Connected, Automated, and Autonomous Forces

Jeff Vu – jeff.vu@navy.mil
Viral Bhalodia – viral.bhalodia@navy.mil
Samia Ali – samia.ali@navy.mil
Johanna Flores - johanna.flores1@navy.mil
Haiqiao Lin - haiqiao@spawar.navy.mil
Daniel Silva - dsilva@spawar.navy.mil
Allen Qiu - aqiu@spawar.navy.mil
Luis Martinez - lmmartin@spawar.navy.mil
James Jen – james.jen@navy.mil
Steve Fogg - steven.fogg@navy.mil
Jose Carreno – Jose.carreno@navy.mil
Dr. Stephanie Hszieh – Stephanie.hszieh@navy.mil

Space and Naval Warfare Systems Center Pacific
53560 Hull Street
San Diego, California 92152-5001

Abstract

In the face of growing global uncertainty, the Navy, led by Chief of Naval Operations Admiral John Richardson, has structured its strategy around a “Design for Maintaining Maritime Superiority” in which the Navy “will make our best initial assessment of the environment, formulate a way ahead, and move out.” Inherent in this “design” is the ability to “continually assess the environment.” In short, the uncertainty that the Navy faces necessitates deliberate course corrections and the use of constantly evolving strategy, tactics, and tools. Central among these tools is the need to have agile command and control (C2) software systems and updates that can be rapidly deployed on any of the plethora of naval systems.

In recent years, SPAWAR Systems Center Pacific (SSC PAC) has invested significant resources into improving the Navy’s software distribution chain through the implementation of the SPAWAR Unified DevOps Orchestration Engine (SUDOE). Traditionally, the Navy has relied on highly trained engineers to perform time-consuming software installations on ships and other duty stations. SUDOE, on the other hand, follows a set of best practices and utilizes a suite of open source and custom built tools for provisioning management, code scanning, code compilation, automated software orchestration, and user friendly infrastructure controls that can be implemented remotely. SUDOE can be tailored and applied to DoD projects of varying requirements and complexity -- ranging from Programs of Records to small internally funded research projects. Implementation of the SUDOE has decreased time and cost, while increasing security and quality

Introduction

At the Fall Defense and Industry Forum, Capt. Kurt Rothenhaus, program manager for the Navy's Tactical Networks Program Office, outlined the importance of DevOps and how it will touch many aspects of design, development and support of applications used by the Fleet. He said that, "With DevOps, security is no longer integrated as a separate, final step, but addressed on a continual basis starting from the initial phase of development through deployment."

SPAWAR systems center Pacific is looking into ways of improving the speed at which software is delivered to the Fleet by using the DevOps methodology and Automation. The tool SUDOE, SPAWAR Unified DevOps Orchestration Engine, was built on top of the DevOps principle and technologies such as Ansible, Kubernetes, Containers and Microservices to look at ways to mitigate the bottleneck between development and production. SUDOE has leveraged containerization technology, which allows applications with everything they need to run (e.g. code, runtime, system tools, libraries, etc.) to be wrapped up into a single software package, to build an architecture leveraging DevOps principles. Containerization guarantees that the application will run in a consistent manner, regardless of the environment in which the container is deployed. A common environment where all applications will behave the same is the bedrock of this paper.

In this paper, we analyze the advantages of using DevOps principles (continuous integration/ continuous deployment), Microservices architecture, automation, use of container technologies and container orchestration tools to develop, deploy and test software in the Navy. To study the advantages, we use a custom tool built on the above concepts called SUDOE.

The Problem Space

In the Navy's C2 Domain, there are hundreds of different environments – from large, well-connected environments running on shore to small networks operating aboard a single ship - where Navy software needs to run. This fact accompanied with every rising cyber security threats makes the area of software deployment extremely complex and challenging. There is a need to have regular software updates that can be installed with minimal delays, and relying on trained engineers makes it very difficult because of cost concerns and the sheer number of environments. In a production environment that software must be bug-free, with consistent and reliable operation. Unfortunately, many Navy software systems are very complex, consisting of a myriad of services spread out across multiple servers.
Seamless flow of code from development, through deployment, to the testing and production is essential for effective software development and the deployment of updated bug free software. A bottleneck in any of these phases leads to delayed schedules, increased cost, and poor performance. Bottlenecks are frequently in the deployment phase in the Navy's challenging environment.

We observe a few common contributing factors: (1) The prevalence of Agile processes and commitment to solid software engineering practices has increased not only code quality but also increased throughput and thereby put enormous pressure on the deployment end of the software development lifecycle to keep pace (2) The necessity to comply with numerous DoD standards, address cyber security concerns, and support different enclaves has slowed deployment drastically.

The Proposed Solution

At SSC Pacific, we have bridged the gap between development and operations (testing and production) and greatly improved the deployment phase by employing an architecture which leverages DevOps services and containerization in conjunction with custom built software tools. As an innovative approach, we use the advantages of containerization to reduce the bandwidth needed to deploy software. Following a set of best practices and utilizing a suite of open sourced software and tools.

Custom built tools for provisioning management, code scanning, code compilation, automated software orchestration, and user friendly infrastructure controls has led to a decrease in cost, on time deliveries, and more secure higher quality products. Our process has been tailored and successfully applied to DoD projects of varying requirements and complexity -- ranging from Programs of Records to small internally funded research projects.

To demonstrate the advantages of using DevOps, Continuous Integration and Continuous Delivery, Microservices architecture and use of Automation in C2 domain and software development for the Navy, we used SUDOE to automate the deployment of a sample application in a container, the sample application was built using Microservices architecture.

SPAWAR Unified DevOps Orchestration Engine (SUDOE)

SUDOE is a government-developed software suite of tools for fast, reliable, secure and automated deployment of software and infrastructure. It leverages technologies such as virtual machines, containers and micro services. It was designed with the goal of delivering software capabilities in a few hours instead of days/months to the warfighter. The tool aims at achieving this by automating the process of orchestration, configuration and installation of software services. The diagram in Figure 1 describes the logical view of SUDOE. The client, either your browser or a command line interface interacts with the SUDOE REST (Representational State Transfer) API (Application Programming Interface) layer which automates the process of orchestration, software installation and configuration, the security aspect is achieved by using PKI authentication to automatically enforce security models. At the infrastructure level, kubernetes provides cluster wise deployments, auto scaling, and load balancing capabilities. The underlying architecture of the tool is captured in Figure 2.

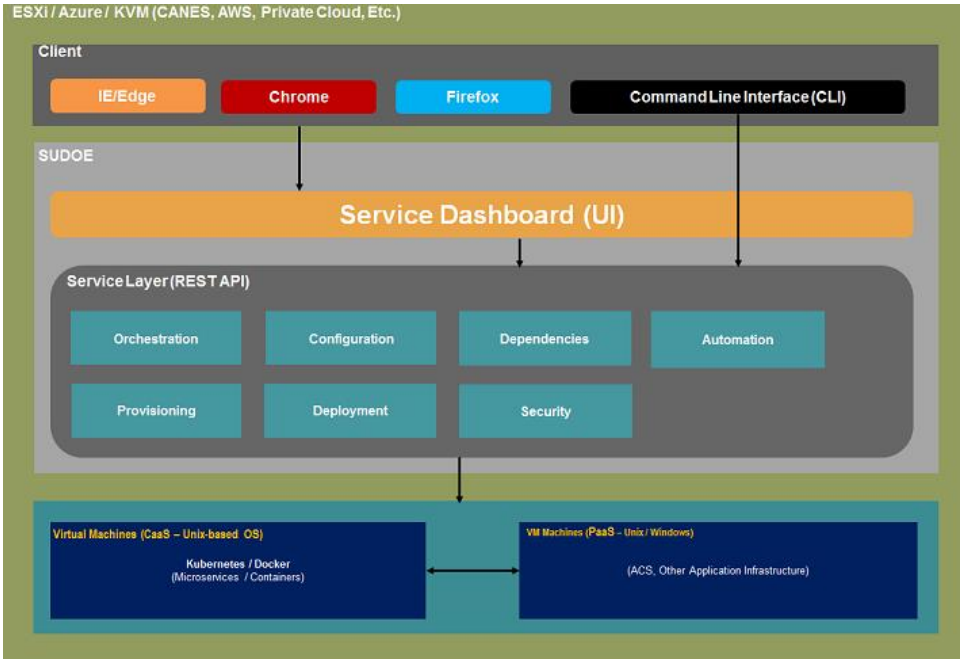


Figure 1 – Logical view of SUDOE

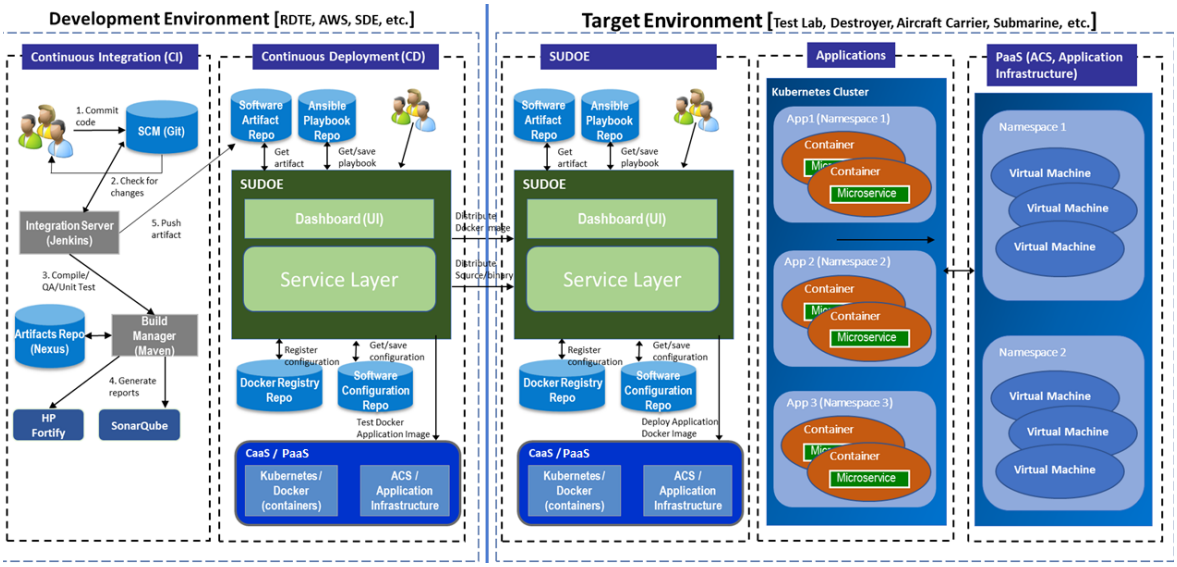


Figure 2 – Architecture diagram of SUDOE

Underlying Technologies – Deep Dive

SUDOE is designed on the following technologies.

Docker Containers:

- Eliminate messy dependency resolution with all-in-one package
- Share kernel with neighbors, less virtualization overhead than VMs, more efficient memory and storage packing

Kubernetes:

- Orchestrate container stack deployments over multiple nodes with service discovery, health-checking, and auto-healing
- Additional rules for ingress traffic from cluster-external clients and egress to downstream VM-based stateful services

Ansible:

- Automate virtual machine installation with declarative playbooks
- Place infrastructure and dependency installation under version control.

Jenkins:

- Automation server popular for building artifacts in continuous integration/continuous deployment pipeline
- Extend familiar environment to support Ansible deployments as next logical step

- Inherit existing server configuration through existing plugins

GIT:

- Place stable Ansible playbooks under source control
- Allows for versioning and rollbacks of virtual machine cluster provisioning

Node:

- Need for unifying API to prioritize individual connections and transactions rather than processing power
- Plethora of additional modules to support multi-transactions across additional APIs

Angular:

- Modular framework enables independent feature development with less integration hassle
- Allows data to be collected across orchestration APIs and collated into a single view of the hybrid deployment

Token Authentication:

- Offload credential presentation to upstream identity provider that issues a short-lived access token
- Token is shared among downstream services, enabling SSO without the need for a shared session cache

Use Cases of SUDOE

To understand the need of using a tool such as SUDOE we look at some scenarios which a typical software development team might face in their day to day activities.

Figure 3, describes how using a tool which promotes automation and DevOps will help reduce the time a new development team member takes to set up their development environment, thereby, reducing development costs and time.

Figure 4, looks at the scenario when a new feature is added to the software and needs to be tested and deployed. Using continuous integration/ continuous deployment it becomes very easy to test the new feature and deploy it in production.

Figure 5 and 6, describes how using SUDOE effects the various stages of the Software Development Life Cycle when compared to not using the tool or similar technologies.

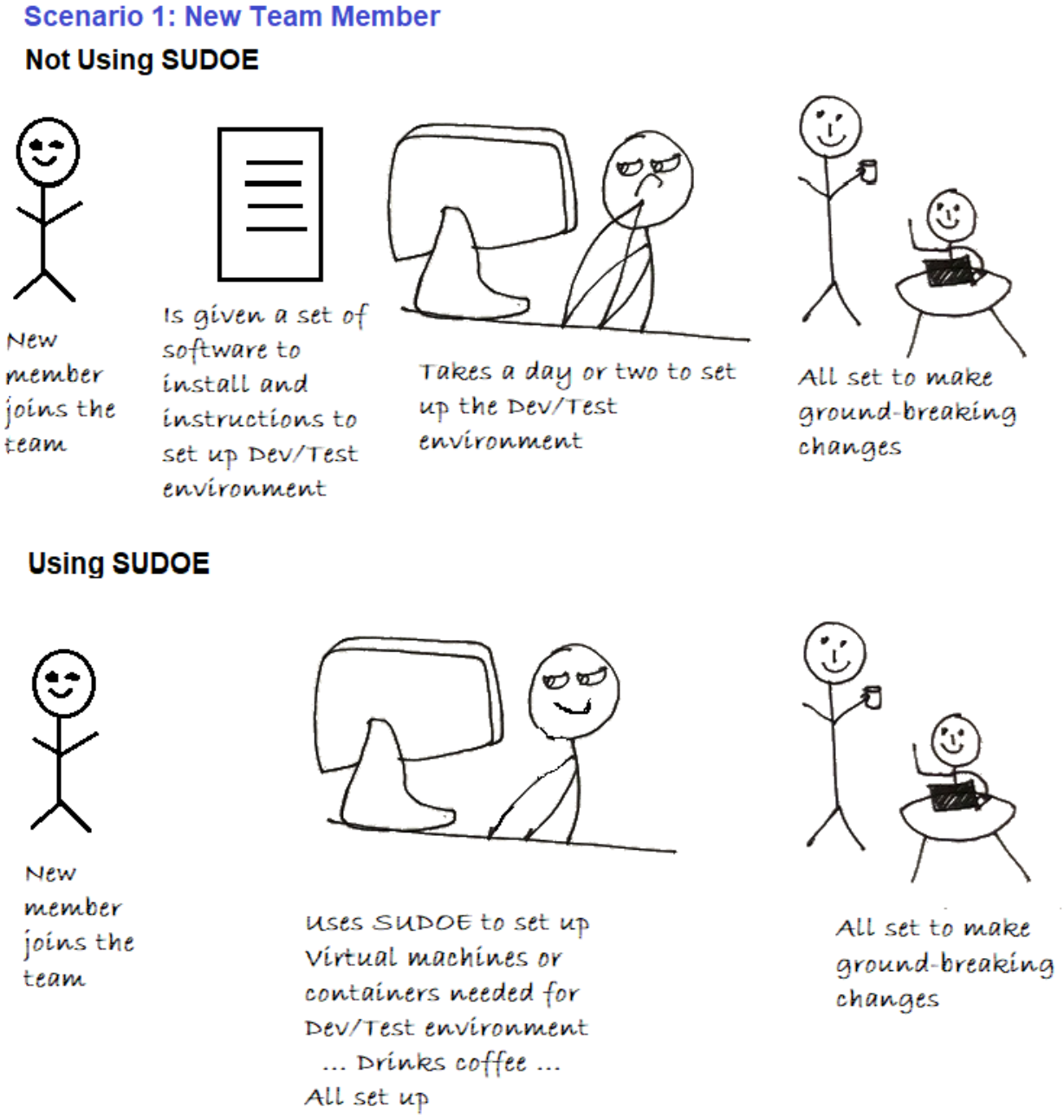


Figure 3 – Compares the events and steps taken to get a new development team member up to speed when using SUDOE and when not.

Scenario 2: New Features to be Tested

Not Using SUDOE

A new feature needs to be tested

Testing environment is prepped for staging, patch is deployed, takes about a day.

Integration Team: Environment ready for testing
Testing Team: About time!
Testing begins ....

Using SUDOE

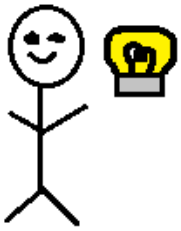
A new feature needs to be tested

Tester logs into the testing namespace in SUDOE, deploys container or virtual machine with latest change .... drinks coffee and testing begins.

Figure 4 – Compares the events and steps taken to test a new software feature when using SUDOE and when not.

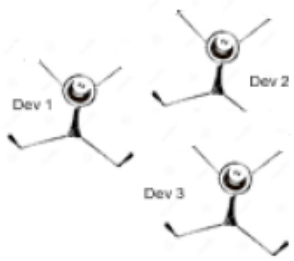
Scenario 3: Software Development Lifecycle

Not Using SUDOE



Now that we have this awesome idea for our app and the software spec lets begin

Devloper 1: What database have we decided on?
Developer 2: Postgres
Developer 3: Roger that.
Lets get started.



Yay! it works in the development environment. Its time for integration.

After a week ....
Yay! we are all set in the Ops side! The customers love it.

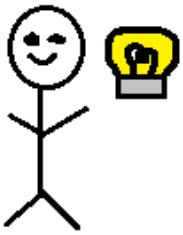
Developer 1: Oh no! I messed up, we need to revert to the previous version, the current version is a no go.

Integrator: Spends a counple of days to get the previous version of the software and reverts to it.

Figure 5 – Describes the Software Development Lifecycle when not using the SUDOE tool.

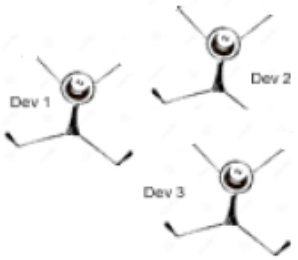
Scenario 3: Software Development Lifecycle

Using SUDOE



Now that we have this awesome idea for our app and the software spec lets begin

Developer 1: what database are we using?
Developer2:just use the container x in SUDOE
Developer3:Gosh,I still get nightmares thinking about our days BS(Before SUDOE)



Yay! t works in the dev environment, its time for integration. Let give the integrators our config file to import into SUDOE

Developer 1: Oh no, I messed up. We need to revert to the previous version.

Integrator: Goes into SUDOE, chooses the previous image tag and hits enter.
A minute later …
Integrator: We are good to go!

Figure 6 – Describes the Software Development Lifecycle when using the SUDOE tool.

The Test Scenario and Results

A test application, captured in Figure 7, which consists of a 3D DDG (Guided Missile Destroyer), displays information about parts of the ship when clicked. Each ship part is written as a separate service, thereby utilizing the microservice architecture. The developer makes a change to one of the services, and pushes their code into SUDOE's GIT repository, the code is built by SUDOE's Jenkins, which also builds a new container image of the code and tests the code for credibility using the SonarQube plugin. The new image is available through SUDOE's frontend. The user can choose to deploy the new image and the change made is reflected almost immediately. If this process was to be done without using DevOps, automation, continuous integration/continuous deployment then to accomplish this task the developer would first have to make changes to the code, push the change to their configuration management and use a build server to build it. Then, it would be passed on to the testing team, who would deploy the application in their test environment, after the tests complete, the integration team would deploy it in production.

Table 1, below displays four scenarios that were carried out to understand the advantages of using these technologies for the Navy. When using SUDOE and its underlining technologies we noticed that the test scenarios took far less time to complete when compared carrying out without using SUDOE.

After comparing the results of running the sample application using SUDOE and without it, we conclude that it is advantageous for the C2 Domain to use DevOps principles, automation and Microservices architecture and applications such as SUDOE which not only improve the speed at which the software can be deployed but also reduce the costs associated with deployments considerably.

SNO	Description	Using SUDOE	Time taken	When not using SUDOE	Time taken
1	A developer makes a change to one of the services in the test application	The developer makes the necessary change in SUDOE's GIT and pushes the change. (Figure 8) SUDOE's Jenkins build's a container image (Figure 10), the Jenkins uses SonarQube to test the code. The new container image is available through SUDOE's web interface. With once click the new change can be deployed and is seen immediately.	5 minutes	The developer makes the change in code, the code is deployed, then send for testing. Once it passes testing the code is deployed in production.	2 days
2	The test application has a compromised service which needs immediate attention	When using the microservice architecture, each service is its own independent entity. The developer pulls down the affected service and fixes it without affecting the application. The other parts of	1-3 days There is no downtime for the customers	If the application is a monolith, the entire application may be compromised when one service or a part of the application is affected. The	1 week The customers don't have access to

		the applications except the affected service are available to the users.		development team will need to stop service, pull down the application, fix the issue and re-deploy.	the application.
3	The user wants to deploy and test new software	The user provides SUDOE with the repo where their scripts to deploy the application live, they use the SUDOE front-end to deploy the application and test it immediately.	A few hours	The user will first need to set up an environment where they can deploy their application. Then they manually install it.	A couple of days.
4	The developer wants to revert to a previous version of the deployed application	The developer uses SUDOE's edit deployment screen and selects the previous deployment from the drop down and deploys and tests the application. (Figure 9)	A few minutes	The developer will have to bring down the application and manually revert to the old state. If there were any configuration changes made, the developer will have to take care of them as well.	A day or so.

Table 1 – Compares the results of using the SUDOE tool and not using it when enacting various scenarios described in column 2.

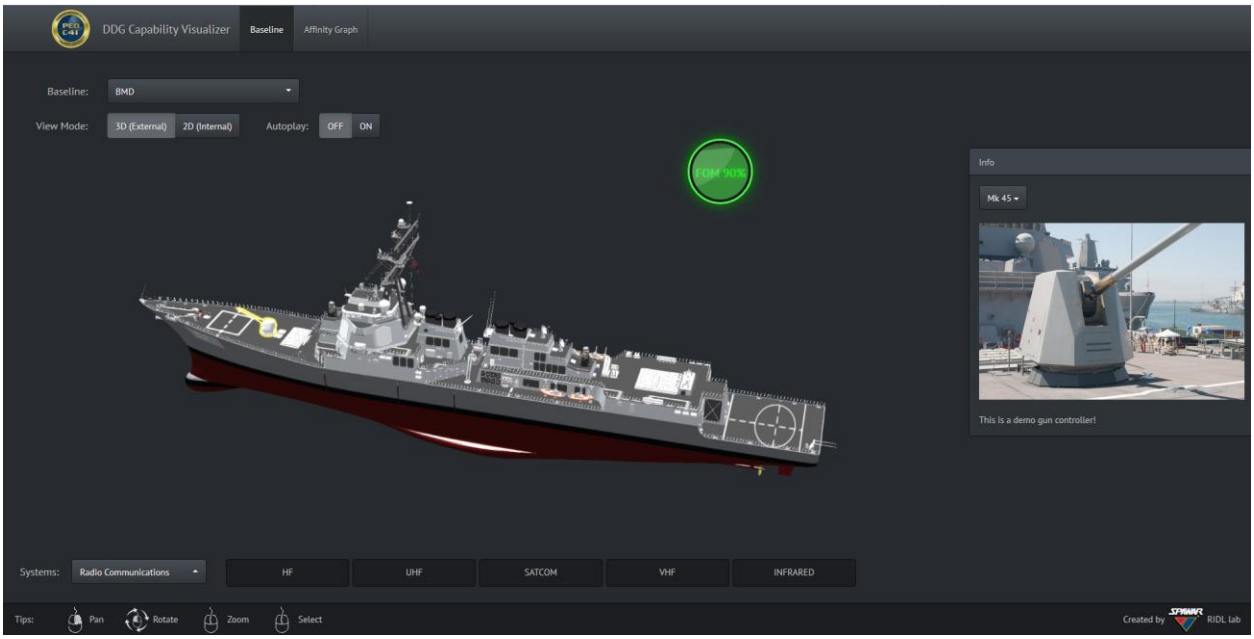


Figure 7 – The 3D DDG test application which displays details about each part of the ship when clicked. The application is written using the microservice architecture.

Figure 8 – Developer makes a change to the gun controller code using GIT.

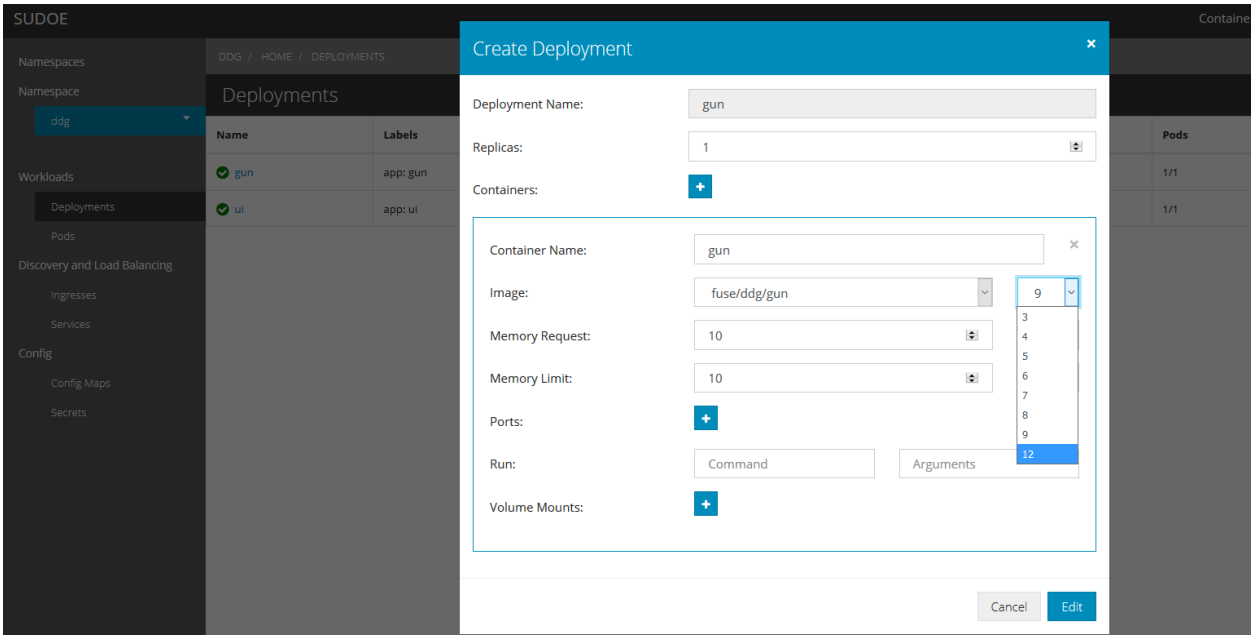


Figure 9 – The Deployment page of the SUDOE tool used to revert the version of the software.

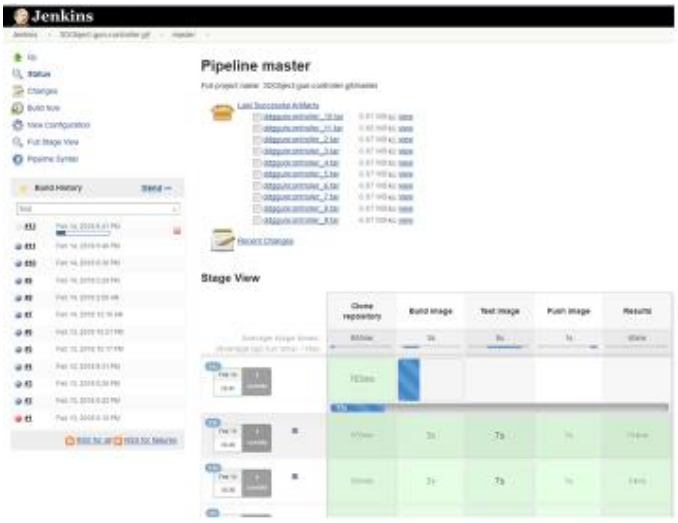


Figure 10 – The Jenkins pipeline view when an image is building through SUDOE.

Technical Challenges

The following are some of the challenges we faced when implementing the SUDOE tool:

- When integrating with existing Public Key Infrastructure, session must be shared among multitude of downstream Application Programming Interfaces (API's)
- Extending container namespace restrictions in kubernetes to Virtual Machines. Since the Navy application still depend heavily on the use of Virtual Machines, we had to go with a hybrid approach

to using Docker Container. The SUDOE tool combined orchestration of software on Virtual Machines and Container to cater to the needs of U.S Defense projects.

- Creating a service to drive Ansible installations, the automation is carried out using Ansible. SUDOE augments Jenkins experience to support Ansible plays as well as artifact builds with additional configurations from its API
- Raising developer awareness of Docker and Ansible. The efficiency of orchestration pipeline is directly proportional to the quality of deployment artifacts maintained by application developers. Therefore, teams require more experience with Docker and/or Ansible development for rapid declarative deployments.
- Uncertainty of STIGS (Security Technical Implementation Guides) required for container images

The Way Ahead

The test scenario and application used in this paper were very simple, in reality, Navy applications are far more complex and consist of many moving parts. The adoption of SUDOE and the other tools discussed in the paper might lead to interesting results when studied with respect to complex applications developed by collaboration of various teams and organizations, each team might have different development strategies and tools used for development, in the future, we plan to test SUDOE on a more complex application.

Conclusion

This paper presents the initial steps taken to incorporate the use of a unique combination of commercially used open-sourced technologies such as Docker container, Kubernetes, Virtual Machines, Ansible, DevOps principle and Microservices architecture in the Navy's Command and control Domain with the aim of improving the speed at which software is made available to our warfighter and thereby reducing costs. To understand the effects of using the above technologies, a proof of concept, SUDOE is used, which is a government-developed tool written using Angular and Node combines the above mentioned technologies to give a one stop solution for Navy application's development, automation, testing and deployment.

The findings of using SUDOE established some very important foundations for the next steps in this effort. They support our claim that use of these technologies - when used with SUDOE, which provided a unique, custom-built Navy tool - made deployment and development much faster and reduced costs considerably.

In addition, the findings supported that there is a lot of scope in automation and software deployment. The U.S Department of Defense and defense organizations elsewhere will be well-served to expand their research and efforts in building tools such as SUDOE as a wrapper above latest commercial technologies to promote automation and continuous deployment and development of their software applications. This would ensure a fast turnaround and throughput of software in the challenging domain of command and control.

References

Matthias, K., & Kane, S. P. *Docker: Up and running*. Beijing: OReilly, 2016.

Space and Naval Warfare Systems command Public Affairs, 'PEO C4I Streamlines Afloat Capability Delivery with DevOps Model', 2017. [Online]. Available: http://www.navy.mil/submit/display.asp?story_id=103054 [Accessed: 25- July- 2018].

Newman, S. *Building microservices*. Beijing: OReilly, 2015.

Kim, Gene, et al. *The DevOps Handbook: How to Create World-Class Agility, Reliability, and Security in Technology Organizations*. IT Revolution Press, LLC, 2017.