



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

**A DEVSECOPS APPROACH FOR DEVELOPING AND
DEPLOYING CONTAINERIZED CLOUD-BASED
SOFTWARE ON SUBMARINES**

by

Bridger A. Smith

December 2021

Thesis Advisor:
Co-Advisors:

Neil C. Rowe
James B. Michael
Kaitlynn Castelle,
Old Dominion University

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE December 2021		3. REPORT TYPE AND DATES COVERED Master's thesis
4. TITLE AND SUBTITLE A DEVSECOPS APPROACH FOR DEVELOPING AND DEPLOYING CONTAINERIZED CLOUD-BASED SOFTWARE ON SUBMARINES				5. FUNDING NUMBERS
6. AUTHOR(S) Bridger A. Smith				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000				8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A				10. SPONSORING / MONITORING AGENCY REPORT NUMBER
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.				12b. DISTRIBUTION CODE A
13. ABSTRACT (maximum 200 words) There are unique challenges for using secure cloud services in disconnected resource-constrained environments and with controlled data. To address those challenges, this thesis introduces a tactical-edge platform-as-a-service (PaaS) solution with a declarative-delivery method for submarine Consolidated Afloat Network Enterprise Services (CANES) operating systems. The PaaS is adapted from the Department of Defense's Big Bang core elements for submarine-focused outcomes. Using the Team Submarine Project Blue initiative as a case study, this thesis consists of a feasibility study for running containerized applications on different submarine-compatible baselines and applying a prototype declarative software-delivery method called ZARF. We demonstrated the feasibility of using ZARF for packaging and automated deployment of the Project Blue PaaS and its software to the submarine CANES infrastructure. This research culminated in successful integration tests on a current and future submarine hardware and software baseline. The thesis documents the execution of the research, lessons learned, and recommendations for the Navy's path forward for development of secure software and declarative deployment in air-gapped environments.				
14. SUBJECT TERMS Development Security Operations, GitOps, containers, Kubernetes, infrastructure-as-code, configuration-as-code, submarines, cybersecurity, air gap, sustainment, platform-as-a-service				15. NUMBER OF PAGES 129
				16. PRICE CODE
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**A DEVSECOPS APPROACH FOR DEVELOPING AND DEPLOYING
CONTAINERIZED CLOUD-BASED SOFTWARE ON SUBMARINES**

Bridger A. Smith
Lieutenant, United States Navy
BS, United States Naval Academy, 2015

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN CYBER SYSTEMS AND OPERATIONS

from the

**NAVAL POSTGRADUATE SCHOOL
December 2021**

Approved by: Neil C. Rowe
Advisor

James B. Michael
Co-Advisor

Kaitlynn Castelle
Co-Advisor

Alex Bordetsky
Chair, Department of Information Sciences

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

There are unique challenges for using secure cloud services in disconnected resource-constrained environments and with controlled data. To address those challenges, this thesis introduces a tactical-edge platform-as-a-service (PaaS) solution with a declarative-delivery method for submarine Consolidated Afloat Network Enterprise Services (CANES) operating systems. The PaaS is adapted from the Department of Defense's Big Bang core elements for submarine-focused outcomes. Using the Team Submarine Project Blue initiative as a case study, this thesis consists of a feasibility study for running containerized applications on different submarine-compatible baselines and applying a prototype declarative software-delivery method called ZARF. We demonstrated the feasibility of using ZARF for packaging and automated deployment of the Project Blue PaaS and its software to the submarine CANES infrastructure. This research culminated in successful integration tests on a current and future submarine hardware and software baseline. The thesis documents the execution of the research, lessons learned, and recommendations for the Navy's path forward for development of secure software and declarative deployment in air-gapped environments.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

I.	INTRODUCTION.....	1
A.	PROBLEM DOMAIN	2
B.	CURRENT INITIATIVES.....	3
C.	APPROACH.....	4
D.	THESIS OVERVIEW	5
II.	PREVIOUS WORK.....	9
A.	DEFINITIONS AND KEY CONCEPTS	9
B.	READINESS FOR MAINTENANCE AND SUSTAINMENT.....	13
C.	SUSTAINMENT SOFTWARE	15
D.	DEVSECOPS SECURE PLATFORM-AS-A-SERVICE.....	18
III.	METHOD	23
A.	PROBLEM CONTEXT.....	23
B.	SOLUTION ASSUMPTIONS.....	24
C.	SOLUTION DEVELOPMENT	24
IV.	APPROACH.....	29
A.	PROJECT BLUE’S PLATFORM-AS-A-SERVICE.....	29
B.	ZARF AIR-GAP DELIVERY TOOL.....	34
1.	Appliance Mode (Single-Use Mission Applications):.....	35
2.	Edge Mode (with Istio Service-Mesh)	38
3.	Data-Center Mode	39
V.	CYBERSECURITY CONSIDERATIONS	41
A.	ESTABLISHING CHAIN OF TRUST IN AIR-GAP DELIVERY	41
B.	ZARF AIR-GAP TRANSFERS.....	42
VI.	RESULTS	45
A.	TESTING.....	47
1.	CANES Lab	47
2.	Sandbox Testing.....	49
VII.	CONCLUSION	51
A.	CONCLUSIONS FROM THE TESTING	51
B.	RECOMMENDATIONS	52

C.	COMMITMENT TO DEVSECOPS	53
D.	SOFTWARE IS NEVER DONE	55
E.	INTEROPERABILITY ACROSS COMPONENTS.....	55
F.	ONE TEAM, ONE FIGHT	56
G.	FUTURE RESEARCH.....	57
APPENDIX A. THE BEAST CORE DIGITAL TWIN AND TECHNICAL DOCUMENT VIEWER		59
APPENDIX B. MASTER JOB FILE		71
APPENDIX C. THE BRIDGER PROJECT		75
APPENDIX D. ZARF		85
A.	PHASE 1: PLATFORM-AS-A-SERVICE INSTALL.....	86
B.	PHASE 2: APPLICATION AND DATA INSTALL	91
LIST OF REFERENCES		101
INITIAL DISTRIBUTION LIST		109

LIST OF FIGURES

Figure 1.	Core elements for Big Bang. Adapted from Platform One Big Bang.	26
Figure 2.	Official Project Blue PaaS logo	30
Figure 3.	High-level architectural view of the services chosen for deployment to the CANES infrastructure	33
Figure 4.	Official ZARF logo	35
Figure 5.	Generalized Project Blue PaaS and ZARF components	36
Figure 6.	Project Blue PaaS services	37
Figure 7.	Big Bang elements in ZARF Appliance mode	37
Figure 8.	ZARF Edge mode services	38
Figure 9.	Additional Big Bang elements in ZARF Edge mode	38
Figure 10.	ZARF Data-Center mode	39
Figure 11.	EFK elements from Big Bang in ZARF Data-Center mode	39
Figure 12.	Process of a ZARF air-gap transfer	43
Figure 13.	Selection screen	59
Figure 14.	Full hull view of USS NEVERSLEEPS	60
Figure 15.	Component selection and specification view	61
Figure 16.	Search function	62
Figure 17.	Search component screen zoom	63
Figure 18.	Flow mode visualizer	64
Figure 19.	Component state	65
Figure 20.	Two-dimensional system diagram viewer	66
Figure 21.	Space directory search	67
Figure 22.	Avatar mode	68

Figure 23.	Avatar mode component selection.....	69
Figure 24.	Avatar path display	70
Figure 25.	Master Job File dashboard	71
Figure 26.	Component and job search.....	72
Figure 27.	Procedure creation	73
Figure 28.	Master Job File reference viewer	74
Figure 29.	Critical steps and inspection points.....	74
Figure 30.	Job status tracking.....	75
Figure 31.	Individual job data	76
Figure 32.	Integrated work schedule	77
Figure 33.	Timeline changes	78
Figure 34.	SOSMIL.....	79
Figure 35.	Nightwork	80
Figure 36.	Print nightwork	81
Figure 37.	Job summary	82
Figure 38.	Refit reports	83
Figure 39.	Print reports.....	84
Figure 40.	Initial ZARF interface.....	85
Figure 41.	Grafana metrics dashboard	94
Figure 42.	Loki metrics dashboard.....	94
Figure 43.	DOOM container	95

LIST OF TABLES

Table 1.	Summary of Project Blue PaaS and Beast Core mission services.....	32
Table 2.	Resources available for submarine CANES	45
Table 3.	CocoWow resource requirements	46
Table 4.	Conservative virtual machine pre-build specifications for CANES lab testing.....	47
Table 5.	Submarine platform integration tests	48
Table 6.	ZARF sandbox testing	49

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

ABMS	Advanced Battle Management System
ACS	Agile Core Services
C2C24	Compile to Combat in 24 hours
CaC	configuration-as-code
CANES	Consolidated Afloat Network Enterprises and Services
CBM+	conditions-based maintenance program
CI/CD	continuous integration and continuous delivery
CNAP	cloud-native access point
CNCF	Cloud Native Computing Foundation
CNO	Chief of Naval Operations
CVEs	common vulnerabilities and exposures
DCAR	DOD Common Artifact Repository
DevOps	development operations
DevSecOps	development security operations
DISA	U.S. Defense Information Systems Agency
DOD	U.S. Department of Defense
DORA	DevOps Research and Assessment
EFK	Elastic, Fluentd, Kibana
eMASS	Enterprise Mission Assurance Support Service
GAO	U.S. Government Accountability Office
GB	gigabyte
IaC	infrastructure-as-code
ILS	integrated logistics support
IT	information technology
IWS	integrated work schedule
JADC2	Joint All Domain Command and Control
JFMM	Joint Fleet Maintenance Manual
K3s	lightweight kubernetes container orchestrator
K8s	Kubernetes
LDS	Logistics Data System

MB	megabyte
MJF	Master Job File
MVP	minimum viable product
NAVPLAN	Navigational Plan
NAVWAR	Naval Warfare Information Systems Command
NIST	National Institute of Standards and Technology
NIWC	Naval Information Warfare Center
N-MRO	Naval maintenance repair and overhaul
NNPI	Naval nuclear propulsion information
NPS	Naval Postgraduate School
NUWC	Naval Undersea Warfare Center
P1	Platform One
PaaS	platform-as-a-service
PEO	Program Execution Office
PLG	Promtail, Loki, Grafana
PrOM	Project Overmatch
QA	quality assurance
RAISE	Rapid Assess and Incorporate Software Engineering
RAM	random access memory
RFC	request for comment
RMF	risk management framework
SCCA	Secure Cloud Computing Architecture
SHA	Secure Hashing Algorithm
SOSMIL	Safety of Ship Maintenance Index List
SRS	Submarine Readiness Squadron
SSBN	ballistic-missile submarine
SWAP	Software acquisitions and practices
TB	terabyte
TLS	transportation-layer security
TRF	Trident Refit Facility
UNNPI	unclassified Naval nuclear propulsion information
USAF	United States Air Force

USG	United States Government
vSOC	virtual security operations center
WAF	work authorization form

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

I would like to give special acknowledgement to my wife, Kalli, and children, Amorie and Everly. I could not have done this without your unwavering love and support.

I would like to thank Dr. Rowe and Dr. Michael for giving me the opportunity to pursue something I am passionate about. My time at NPS and this thesis has been a truly challenging and rewarding experience because of your investment and guidance.

To Dr. Kaitlynn Castelle, Mr. Bill Baker, Mr. Dan McMath, and the Project Blue Team, thank you for believing in me and for going above and beyond to guide and support my vision. It has been the highlight of my career.

To Matt, Sarah, Wen, Christian and the Beast Core team, thank you for supporting the initial mission application design and development. Because of your efforts, I truly believe software is never done, and in fact never sleeps.

To Rob, Jeff, and the Defense Unicorns Team, thank you for flying to Rhode Island in a hurricane to support this work. The energy you bring to this project and the culture you have created are inspiring.

Lastly, to the people who I connected with in support of this research, who generously offered their time, perspective, and feedback over the course of this study, across all branches, uniformed, civilian, and contractors, I am humbled and truly grateful for your time and willingness to help find and build solutions to improve the lives of myself, and the other warfighters and maintainers working today.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

The United States Navy's submarine crews currently plan and execute ship maintenance using Microsoft Office Excel spreadsheets. Sailors regularly use Excel's fill function to manually paint cells to create a Gantt-chart workflow for the ship's maintenance deconfliction plan or master integrated work schedule (IWS). This schedule is just blocks in the spreadsheet, merged and stacked, for tracking maintenance activities, with bars color-coded for the department responsible for completing each task. Building the IWS and updating it as the maintenance environment changes provides a common operating picture for a submarine crew and a Trident Refit Facility (TRF) during maintenance. For both stakeholders, the goal is to plan, coordinate, and execute timely and efficient maintenance to get the ship to sea on schedule.

While Microsoft Excel is authorized and readily available, it cannot be customized, adequately handle concurrent reads and writes, or be easily updated when the inevitable change to the schedule occurs. Using Excel requires manually and serially assembling job information from users in different submarine departments and from the disconnected ship and shore information technology (IT) systems. Given these challenges, the IWS is insufficient for managing maintenance, and manual routing of paperwork based on the IWS, such as the daily work authorization list called "Nightwork," is also required. Following this approach to maintaining a common operating picture is tedious, error-prone, and subject to revision-control and administrative-bottlenecks issues, suggesting that other technology solutions should be pursued.

Partial digitalization of traditional processes to achieve desired outcomes is inefficient and loses time used for problem-solving, resource-contention resolution, and meaningful collaboration. These three elements are of high importance for the time-compressed cycles of submarine maintenance, where a several-hour delay could mean the difference between success and failure for the vessel and the fleet schedule. What is needed is a solution which eliminates the technical and cultural limitations that hurt efficiency in the fast-paced submarine refit environment. Automated processes, co-development of software with the TRF maintainers and submarine operators, and a secure deployment

pipeline could replace ad-hoc solutions, providing maintainers with a more interconnected and adaptable environment for getting a ship underway.

Culturally, more attention should be paid to the system requirements given by submarine users. Since inception, the focus of the Trident Logistic Data System (the holistic logistic solution for ballistic-missile submarines) and other modernization strategies focused on the needs of shore facilities, leaving sailors forced to use poorly fitting tools like Excel. Considerations of ship interfaces for collaboration and integration with shore systems have been inadequate. Given that all work on a ship is supported and coordinated by the submariners, it is worth exploring the infrastructure and software delivery limitations for ways to improve maintenance coordination. This will be relevant as ship and shore systems are modernized to support the submarine operational cycle.

A. PROBLEM DOMAIN

For digital modernization, the current operating environment and available technology must be understood and updated to meet new challenges. The capabilities and actions of threat actors operating in cyberspace have become more sophisticated and pose a real threat to national security. This means that cybersecurity must be aggressive when modernizing legacy infrastructure and processes, and in developing software for new mandated strategies (Department of Defense [DOD], 2021b). The United States government (USG) has adopted a proactive way to manage cyber-related risks by focusing on security issues early and more continuously in the software-system life cycle. The framework is called Development Security Operations (DevSecOps). As modern weapon systems become increasingly software-centric and network-oriented (Government Accountability Office [GAO], 2018) DevSecOps has become increasingly appealing to the DOD. To organize these efforts, the USG has issued policy on hardening its cybersecurity posture, promoting software development practices from industry, and reducing the barriers to access of cloud technologies, which should result in a reduction in the software deployment timeframe to warfighters (White House, 2021; DOD, 2019a). With cloud-based architectures, agile development methodologies, and robust cybersecurity

technology becoming more widespread in the DOD, the submarine community can accelerate delivery of modern capabilities.

The planned future of DOD software engineering is to use DevSecOps methods on existing hardware, patch vulnerabilities quickly, and work with users to create software to meets their needs (Rosenburg, 2019). The solutions developed under the DevSecOps initiatives will be a holistic approach to cybersecurity that allows teams to build upon other services, creating a culture that learns from failure and does not repeat the same mistakes. The shift needs to occur as soon as possible because the current processes are outpaced by technology change and increasingly complex cyber threats. Supporting this approach, in 2019 the Defense Innovation Board for Software Acquisitions and Practices (SWAP) concluded the current DOD software acquisitions and development approach is broken and poses an untenable risk to national security (McQuade et al., 2019). It provides examples of the DOD's lackluster record of deploying software-intensive systems on time, on schedule, and with the required performance. Adoption and implementation of DevSecOps could improve DOD software practices.

B. CURRENT INITIATIVES

Consistent with recent policy changes, new programs such as the U.S. Navy's Project Overmatch and the U.S. Air Force's Advanced Battle Management System (ABMS) under the overall Joint All Domain Command and Control (JADC2) initiative, indicate that the DOD will no longer accept years-long software-delivery timelines and frequent program failures. Under new DevSecOps practices, such as building containerized applications and using cloud services, success stories are more common. However, new solutions must be compatible with the existing infrastructure to avoid impeding mission accomplishment.

For the ballistic-missile submarine (SSBN) maintenance community, this means integrating with the Trident Logistic Data System (LDS) and the Consolidated Afloat Network and Enterprise Services (CANES) submarine infrastructure. DevSecOps and cloud solutions for submarines must operate both connected and disconnected to broad-area networks, and from pier-side to surfaced and submerged at sea. New solutions must

work in the resource-constrained and network-constrained environments in which sailors operate and must be designed for their skill levels.

Interviews done as a part of this research with the Navy’s Project Overmatch and the Air Force’s Platform One teams indicated that neither Agile Core Services (ACS) (the Navy software-security offering using RedHat OpenShift Linux) or Big Bang (the Air Force’s software-security service) could currently meet the resource constraints to deploy containerized software on submarines. For ACS, the overhead for processor, memory and storage using OpenShift templates make containerized applications too large to use on current submarine hardware and software baselines. A slimmed version of ACS adequate to run on the submarine platform would require time, budget and technical resources be redirected from the U.S. Navy Surface Fleet DevSecOps pipeline.

Platform One’s Big Bang, “software factory in a box,” offers an open-source tool accredited by the U.S. Defense Information Systems Agency (DISA) for tailoring a DOD-compliant DevSecOps observability platform-as-a-service (PaaS) and software container orchestrator. A PaaS typically includes the analytics, system management, and security processes for deploying applications in a cloud environment to meet observability goals (Azure, 2021b). Observability is essential to DevSecOps because interfaces allow for continuous monitoring of system-performance metrics such as processor and memory usage, as well as logging and alerts to help identify and recover from cyber incidents. It initially appeared that Big Bang’s system monitoring and logging software had processor and memory requirements designed for a large-scale data-center architecture. However, modified Big Bang deployments on an F-16 and U-2 aircraft demonstrated success in resource-constrained environments (Krazit, 2021; Sirota, 2021).

C. APPROACH

For this study, Big Bang was chosen as a starting point due to its cybersecurity capabilities, its comparatively small resource requirements compared to Agile Core Services, its vendor-independent approach, and its core components compatible with deployment on the submarine CANES infrastructure. Following analysis of requirements, Big Bang provided the best scaffold for necessary changes to produce a submarine-

compatible PaaS. In collaboration with the Project Blue team (a submarine-focused innovation and modernization effort supporting Team Submarine), a lightweight PaaS called CocoWow was developed to run containerized software on CANES delivered by an air-gap transfer. Air-gap or disconnected software delivery is standard for submarines consistent with the National Institute of Standards and Technology (NIST) and the DOD Risk Management Framework (RMF).

To deploy across the air gap, a declarative software delivery tool ZARF was created. ZARF packaged the lightweight PaaS and Beast Core container and automated their deployment on the chosen submarine CANES hardware and software baselines. Beast Core is an unclassified three-dimensional model and technical-documentation viewer for systems that are like the ones on a submarine. In general, expected benefits occur for including this type of digital-twin technology into sustainment. ZARF was designed as a declarative delivery model to (1) make installation user-friendly for users without software-developer skills, (2) deploy quickly to shorten the sometimes days-long shipboard software updates and installations, and (3) avoid further hardware and software baseline drift of the submarine CANES baseline which complicates DevSecOps practices at scale. ZARF improves on open-source air-gap tools like Sync offered at Platform One (Repo1 Sync, n.d.). What Sync does with heavy user involvement, ZARF automates in minutes with a few hundred lines of code. For CANES installation and testing, ZARF automated installation and enabled a successful deployment of cloud-based containerized applications for submarines.

D. THESIS OVERVIEW

For the case study done in this thesis research, the author was embedded in Project Blue as a team member to help with several software tasks:

- A deployment feasibility study of a hardened container for the commercial off-the-shelf digital-twin product called Beast Core.
- Development of maintenance-planning support prototypes, for which the author supported user-centered design and wireframing from concept to high-fidelity prototype.

- Iterative development, cybersecurity review and testing of both a declarative software delivery tool and a submarine-focused PaaS.

Following the reciprocity agreement dated November 2020 between the Navy's Chief Technology Officer and Air Force's Chief Software Officer, the Navy could use the Platform One DevSecOps pipeline to certify the Beast Core container. This three-dimensional model was developed consistent with the Model-Based Product Support initiative of the DOD which aims to include three-dimensional modeling in sustainment and maintenance practices (SEA06L, 2019). Using the open interface of Beast Core, Project Blue developed and certified prototypes for submarine maintenance, sustainment, and logistics operations.

Two interfaces for Beast Core are the Master Job File (for the creation and future reference of work packages used onboard a submarine) and The Bridger Project (a maintenance planning and workflow tool providing a common operating picture between the TRF and the shipboard crew). Both services are currently hosted in the Iron Bank under the Project Blue container and can be rapidly updated while maintaining certification. The Iron Bank is a DOD Centralized Artifacts Repository (DCAR) which holds hardened container images with reciprocity granted across components and classification levels (Chaillan, 2020). Within the Iron Bank, the Beast Core and Project Blue containers undergo continuous monitoring for configuration changes and are continuously scanned for common vulnerabilities and exposures (CVEs). More information on these three products is in Appendices A, B, and C.

This thesis begins with an overview of the motivation and designs of the maintenance-focused Project Blue software prototypes. The thesis then covers (1) the design for a lightweight submarine-compatible PaaS, (2) details about the declarative software delivery tool ZARF, and (3) perspectives of working within an agile software team. Chapter II covers current applications for submarine maintenance and potential opportunities for modernization. Chapter III details submarine-specific requirements for lightweight services. Chapter IV introduces the changes to Big Bang to create a submarine PaaS and the declarative air-gap transfer tool ZARF. Chapter V discusses cybersecurity

issues and methods used in ZARF air-gap transfer of software. Chapter VI reports integration tests for the Project Blue PaaS using ZARF at the Naval Undersea Warfare Center CANES test lab in Newport, Rhode Island, the lessons learned from deploying an Iron Bank container, and data-management considerations from deploying a digital-twin application on a submarine CANES baseline. Chapter VII highlights the major accomplishments of this work and suggest areas for future research.

THIS PAGE INTENTIONALLY LEFT BLANK

II. PREVIOUS WORK

This section provides background on the current state of software readiness for United States Navy submarines and their sustainment. As the DOD pursues new solutions and technologies for submarines, it has addressed digital modernization and cybersecurity threats by adopting several industry “best practices.” These include agile-software methods, cloud computing, and DevSecOps. We discuss these trends and their applicability to the U.S. submarine force.

A. DEFINITIONS AND KEY CONCEPTS

This section defines key terms for the remaining chapters of this thesis.

Containers

Containers enable software to run in different computing environments. They are packages of software that wrap all code, dependencies, and settings for an application into a single fully executable unit (Docker, n.d. b). Containerization allows an application and its components (e.g., databases, configurations, and default settings) to be isolated and abstracted from the outer computing environment so they can run under different operating systems and different versions of those operating systems. This provides flexibility to deploy applications organization-wide on any infrastructure, including the cloud, and reduces the risk of depending on a single service provider.

Microservices

In a microservice-based software-system architecture, the system is decomposed into many applications that each accomplish processes (Larson, 2014). The applications communicate by well-defined applications programming interfaces and each is fully abstractable from the rest of the system. Each microservice can have separate process flows, logic, data-access layers, and codebases. Microservices are an alternative to the traditional monolithic architectures of software systems.

Container Orchestrators

Container orchestrators automate the deployment and management of containers on any infrastructure (RedHat, 2019). This removes the need to redesign or reconfigure an application to deploy it in a different environment. Container orchestration also enables virtualization and scaling of microservices such as storage, networking, and security, which are components of cloud-based applications. Management tools can configure containers based on declared states which indicate how the containers should run.

Infrastructure-as-Code

Infrastructure-as-code (IaC) is the concept of managing the infrastructure required for containerized applications. It focuses on the servers, devices, and networking components. Instead of traditional methods to document procedures, infrastructure-as-code automates the configuration of infrastructure in designated files (Terraform, n.d.) describing in code how the infrastructure will be segmented and what resources are required. The result is a declarative repeatable environment that is effectively managed as a single piece of software.

Configuration-as-Code

Configuration-as-code (CaC) is another way to manage the configuration of applications (Macvittie, 2020). Like infrastructure-as-code, it results in a declarative state that can be automated, repeated, and managed as a single piece of software, with a focus on how the applications or services interact with each other on the infrastructure. The combination of infrastructure-as-code and configuration management through configuration-as-code using containerized software creates a declarative state.

Kubernetes

Kubernetes (K8s) is an open-source container-orchestration platform that automates the deployment, scaling, and management of containerized software (Kubernetes, n.d.). It is infrastructure-independent, meaning it can be deployed and executed independent of hardware and software (e.g., in the cloud, on your laptop, or on

an air-gapped server). Automated provisioning of this infrastructure and the deployment of Kubernetes can be done by infrastructure-as-code or configuration-as-code methods.

Platform-as-a-Service

A platform-as-a-service (PaaS) simplifies the management of services and makes deployment of applications on cloud infrastructure easier (Mell & Grace, 2011). The complexities of these processes are abstracted for the user and made more observable with user interfaces. The services provided by the platform range from actual hardware in a data center to software-development tools. A platform-as-a-service typically includes the analytics, system management, and security processes for deploying applications in a cloud environment to meet observability goals (Azure, 2021b).

Continuous Integration / Continuous Delivery

Continuous integration and continuous delivery (CI/CD) shorten the lead time between engineering and operations by automating security, compliance, and verification controls as much as possible while preserving system reliability. Their methods can also be instrumented to track metrics related to software flow and delivery.

DevOps

DevOps (development operations) synchronize two parts of an organization, the development and the information-technology-operations teams, towards common objectives. Traditionally, developers create new features, responding to user feedback and demand. Information-technology operators maintain the stability and consistency of the production environment. In DevOps, these teams cooperate in a tight feedback loop to deliver information-technology services continuously and quickly with high quality and reliability (Azure, 2021a). This feedback loop encourages continuous improvement, contributes to reduced technical debt, and improves the quality and reliability of the information-technology services delivered.

DevSecOps

In DevSecOps (development security operations), security issues are addressed at each phase of the software development life cycle (IBM, 2021). It extends DevOps by including the way software is accredited and secured. If the security of software design and delivery is insufficiently considered until late in software development, there will be increased costs to rework and retest previously completed efforts.

Air gap

An air gap is a cybersecurity practice for improving a system's security or defense against cyber events. It is defined by NIST in RFC 4949. Air-gapped systems are disconnected from the Internet and untrusted networks. Critical sectors such as national defense, payments and banking, energy, and aerospace use air-gapped systems for highly sensitive data or transactions. However, air-gapped systems present a challenge when they must exchange data with other systems, such as during system updates, since they must use portable storage media. Older air-gapped systems lack run-time monitoring, making it difficult to verify whether a system follows cybersecurity policy during the data transfer.

GitOps

GitOps is the process for automatic infrastructure updating for DevOps by dynamically updating it with code-configuration files based on events in the DevOps workflow (such as version controls, collaboration, and merge requests). GitOps changes the declarative state of the infrastructure based on the activity of the real-time system and allows elastic infrastructure management based on demand (GitLab, n.d.). This reduces the workload of building infrastructure and integrates infrastructure operations in the DevOps cycle.

Iron Bank

DOD's Iron Bank repository contains fully accredited container images that are deployable on any infrastructure. Iron Bank provides free, open-source and commercial off-the-shelf software for DOD customers. It also hardens and secures images for consumption (Platform One, n.d.).

Big Bang

Big Bang provides DOD software teams or programs with secure and customizable DevSecOps environments. Big Bang is managed by the U.S. Air Force Platform One team and provides both infrastructure-as-code and configuration-as-code solutions. It also includes a continuous authority to operate (Platform One BigBang, n.d.).

JADC2, ABMS, and Project Overmatch

The DOD's Joint All Domain Command and Control (JADC2) effort tries to connect "sensors to shooters" across platforms, domains, and services. It integrates the array of disparate, often service-specific, command-and-control networks by providing a framework for "data-driven warfare" (Garamone, 2020).

The Advanced Battle Management System (ABMS) is the primary digital modernization project of the U.S. Air Force and supports the JADC2 initiative (Pope, 2020). It tries to establish a common extensible warfighting network with a seamless flow of data across domains and classification levels.

Project Overmatch (PrOM) is the Navy part of JADC2, developing a Naval Operational Architecture that can support distributed maritime operations (Shelbourne, 2020). It provides connections between networks, integrates emerging technologies, and provides applications for both distributed maritime operations and multi-domain operations.

Project Blue

Project Blue is an innovation effort focused on the digital transformation and design for sustainment goals of Team Submarine. Project Blue focuses on giving sailors and maintainers at the Trident Refit Facilities, Submarine Readiness Squadrons, shipyards, and on submarines the tools to keep pace with the digital modernization.

B. READINESS FOR MAINTENANCE AND SUSTAINMENT

Fleet readiness is a fundamental naval objective, characterized as having the manning, training, and equipment necessary to

deploy forward and win in day-to-day competition, in crisis, and in conflict... [and] consistently deliver maintenance on-time and in full, refurbish our critical readiness infrastructure, master all-domain fleet operations, and exercise with like-minded navies to enhance our collective strength. (Chief of Naval Operations [CNO], 2021, p. 7)

Fleet readiness for shipyard tasks and its impact on operational availability are important to achieve these objectives. Based on a recent U.S. Government Accountability Office (GAO) study, 38 of 51 (75 percent) aircraft-carrier and submarine-maintenance periods were completed late from 2015–2019 (U.S. Government Accountability Office [GAO], 2020b). This caused 7,424 days delayed. On average, shipyards were 113 days late for aircraft carriers and 225 days late for submarines, with the two main causes being unplanned work identified after maintenance planning was done and shipyard-workforce performance and capacity.

Similar sustainment issues are occurring across the Navy. The naval aviation community is seeing supportability issues with its aircraft maintenance and logistic software, which significantly reduce readiness (Wilson et. al., 2020). Under Marine Corps Order 4151.22, the Conditions Based Maintenance Program (CBM+) is mandated for Marine Air Ground Task Forces to include platform sensor data and predictive-maintenance analytics into sustainment. CBM+ is prioritized to “increase asset availability” and “reduce life cycle costs,” which should enable a more lethal and ready Marine Corps (Commandant of the Marine Corps, 2020). Common themes such as current software hindering the maintainer’s efficiency and a push for data-driven concepts in sustainment have arisen in the submarine force. However, without effort and investment, resolving from the current readiness issues will take years (GAO, 2020a).

For the Navy to address the performance and capacity issues in the dynamic shipyard environment, maintenance must become more efficient and well-coordinated. This requires developing a common operating picture with stakeholders and providing tools to satisfy the needs of maintainers. Achieving these goals includes addressing the information systems as a major contributor to productivity (Eversden, 2020). In January 2021, the Assistant Secretary of the Navy launched Operation Cattle Drive, calling for upgrading several legacy logistics and financial information systems, due to being

“unnneeded, obsolete, unproductive, insecure and un-auditable” (Department of the Navy Chief Information Officer, 2020). Infrastructure and software modernization is needed for fleet readiness. Furthermore, if the DOD modernization comes only from the acquisition of monolithic commercial off-the-shelf technology, it can take years to deliver and fully implement. It will take even longer to determine if the solutions improve the issues identified by the GAO about the inability to coordinate and execute work.

Besides the physical sustainment of warships, the fleet struggles to adequately maintain and update its software. This is reflected in reports on the submarine force’s cybersecurity practices and the years-long cycles between operating-system upgrades and software overhauls (Ziezulewicz, 2021; PMW-160, 2020). As weapon systems become more software-centric and software better follows industry best practices, the Navy must upgrade its practices. Decades of mandated traditional practices, risk-averse bureaucracies, vendors locking in use of their software and data, and issues in coupling an already stressed system with complex commercial-software requirements, may cancel out the expected benefits of better logistics software.

For readiness, timely maintenance of physical and software systems enables more days at sea and better deployment of leading-edge technologies against the adversary. For the submarine community, progress towards the Chief of Naval Operations’ (CNO) readiness objectives and the (DOD) modernization efforts begin with the modernization of information technology for the TRFs and Submarine Readiness Squadrons (SRS).

C. SUSTAINMENT SOFTWARE

To improve readiness, the Columbia submarine program adopted a Design for Sustainment acquisition strategy which affects the submarine design for maintainability and builds the support for refit execution (Schafer & Baker, 2019). Much as DevSecOps integrates cybersecurity requirements early in the software life cycle, sustaining a ship must also be considered during the initial conceptualization and design. For physical components such as pumps, valves, and pipes, detailed physical-arrangement reviews can ensure access for in-place maintenance and removals. Arrangement reviews can be done with three-dimensional modeling or physical mockups. Overall, a sustainable design

requires effort and a willingness to balance the requirements of people, time, cost, weight and volume, and other resources that occur in submarine design.

Designing software for sustainment is not simple. Software development can be more of an art than a science, with success relying on human elements that are difficult to control. Across the Department, thousands of digital systems and databases, and fit-for-purpose business processes, exists from digitizing the original processes. Each system differs in the degree of documentation, further complicating matters. Also, planning decades-long software sustainment is difficult when technology evolves more quickly than policy and acquisitions. This means hardware or software in a ship may be obsolete by the time the ship is delivered.

Project Blue was founded to explore opportunities to digitally transform sustainment of the strategic submarine fleet and enable squadrons to maintain two SSBN classes at the same time. It proposed that the transition period from the Ohio to the Columbia class SSBN platform could be done at lower risk by improving the efficiency of maintenance support. Process improvements included scheduling and planning, coordinating, executing refit periods, quality assurance, work accounting, and configuration management. Project Blue's ongoing feasibility studies with SRS members helped identify and understand the unique cybersecurity and infrastructure requirements of submarines. They also helped in identify missing capabilities on the Ohio and Columbia platforms and assess where software could improve user efficiency. Three-dimensional modeling, cloud-collaboration tools, and digital threads enable the holistic life cycle view needed to modernize legacy processes.

Project Blue's efforts focuses on submarines, complementary to the existing and planned systems within the Navy. The Naval Maintenance Repair and Overhaul (N-MRO) system, being developed for the Naval Surface Fleet under U.S. Fleet Forces, could also be used by the submarine force. Its maintenance planning and execution software has been designed based on user feedback sessions from Navy Surface Fleet sailors and is linked to the Joint Fleet Maintenance Manual (JFMM), the main reference for Navy maintenance. The Fleet Forces have contracted with Lockheed Martin and its partners, including manufacturing-work and cloud-applications developer IFS, to provide commercial off-the-

shelf software for naval ships and aircraft (DeRosa, 2021). The N-MRO solution is a large-scale replacement for many current practices of the surface fleet and could be adapted for the submarine fleet.

While IFS offers many necessary services for manufacturing work at the TRFs, its delivery timeline relative to the Columbia class is an issue based on how intimately linked the SSBN mission is to the Trident Logistic Data System (LDS). LDS set a precedent by including holistic integrated logistics support (ILS) for the submarine platform. It is a shore-based automated information system, with an environmental-support system and application processes. It is part of logistic support for the Trident Submarine System, to “provide an integrated information system necessary to support the intensified level of maintenance and logistics support required for Trident submarines to achieve their high level of operational availability” (Hiza, 1982). ILS helps identify life-cycle costs and reduce cost growth by ensuring that support, equipment, and systems can meet expected operational needs. From its perspective, systems cannot satisfy operational requirements unless they can be restored to service within deadlines. From its inception, the Ohio program focused on the concept of total ILS and designed LDS to meet the high operational demands and unique maintenance control requirements for Ohio submarines. As the program of record for SSBN logistics, its continued success came from a culture of continuous improvement.

The Columbia program’s Design for Sustainment also embraces the total ILS concept. Like the Ohio program, it will provide necessary capabilities for the Columbia class throughout its life cycle, including integration of planning and production toward the timely completion of submarine refits and incremental overhaul. This system should be compatible with the maintenance activity’s infrastructure and the need for reliability without disrupting the active planning and production work. DevSecOps methods should quickly prototype, test, and iterate new capabilities for this environment and provide confidence in their ability to meet operational objectives.

Planning future operations and sustainment of a major weapon system involves evaluating product-support alternatives and minimizing the logistics footprint (DOD, 2019b). An important product-support activity “Reset” restores systems to desired levels

of capabilities for the unit’s future mission. Embedded reset teams help the operational maintainer, meet requirements for maintenance and operations, improve deployment time, and reduce costs. An example of a “Reset” is a Refit period in which teams do resupply and maintenance to reset the submarine platform for another patrol.

Currently, submarine crews plan maintenance using paper-based methods or using customized Excel spreadsheets. This is because of the mandate of legacy information systems, lack of advocacy for the sailor’s needs, software-acquisition barriers, and a secretive approach to submarine data. Hand-written Gantt charts and manually updated whiteboards are still standard for communicating shipboard, divisional, and daily maintenance responsibilities. Efforts to use existing commercial off-the-shelf solutions such as Microsoft Project, Smart Sheets, and Excel plugins are attractive because of their accessibility and familiarity. However, customizing such solutions for the submarine fleet entails hidden costs as it is risky to rely on products not specifically designed for them. This typically results in software that slowly diverges from the needs, increasing the gap to be closed (Schwartz, 2017). For instance, while Microsoft Project appears to offer a low-cost opportunity to improve submarine-to-SRS collaboration, it has licensing costs and challenges, unnecessary features for man-hour and cost estimation, scaling issues when managing many jobs, irrelevant features for submariners, and a need to provide custom training for crews and maintainers, all of which have limited value for submarine maintenance. A tailored solution could likely deliver better value to this task. Within the submarine force, the Nosis program’s successes in supporting applications for submarines and its sailor-defined use cases does offer lessons learned and collaboration opportunities (Nosis, 2021).

D. DEVSECOPS SECURE PLATFORM-AS-A-SERVICE

Emerging cybersecurity threats have encouraged the DOD to adopt industry best practices (DOD, 2019a) including DevSecOps. These include practices of DevOps, agile methods, and hardened cybersecurity postures. They also include DevSecOps “playbooks” and frameworks, cloud technologies, open-source tools to avoid vendor dependence, GitOps, IaC and CaC services to run K8s architectures, and DevSecOps pipelines for CI/

CD. The Project Overmatch team at the Naval Warfare Information Systems Command (NAVWAR), the Air Force's Platform One team supporting ABMS, the Army Futures Command's software factory, and the Navy PEO-Digital's Black Pearl team have all use DevSecOps environments. These support a baseline PaaS to provide the necessary monitoring, logging, container orchestration, cloud-security controls, and software-development tools to comply with the DOD, DISA, and NIST requirements about cloud and container security:

- DOD Cloud Computing Strategy
- DISA Cloud Computing Security Requirements Guide
- DISA Secure Cloud Computing Architecture (SCCA)
- National Institute of Standards and Technology (NIST) Cybersecurity Framework
- NIST Application Container Security Guide
- DISA Container Hardening Process Guide
- DOD Enterprise DevSecOps Strategy Guide, Version 2.0.
- DevSecOps Tools and Activities Guidebook, Version 2.0.
- DOD Cloud Computing Security Requirements Guide
- DISA's Secure Cloud Computing Architecture (SCCA) for the Cloud environment

Project Overmatch has partnered with RedHat to expand the Navy's deployment of Agile Core Services (ACS). ACS operates at the platform level on the CANES network and allow the different systems on a ship to "speak the same language" (Howard, 2017). With these services, application developers have confidence that data flows correctly and resource redundancies are minimized (Underwood, 2020). RedHat OpenShift templates are

one element of the Navy's baseline DevSecOps pipeline (Naval Information Warfare Center [NIWC], 2020).

DOD Platform One provides an open-source DevSecOps solution called Big Bang (Platform One Big Bang, n.d.). Platform One offers DevSecOps services certified for DOD systems under DISA accreditation. Programs can download and manage their own development platforms using Big Bang or can have the Platform One team manage their application development, testing, and production needs on a Cloud Native Computing Foundation (CNCF) K8s platform called Party Bus (Platform One Party Bus, n.d.). These products use open-source tools to avoid vendor lock-in and enable accreditations by the Iron Bank.

The Army Futures Command and Army Software Factory partnered with VMware to use its Tanzu product for its standard software-security baseline (Patel, 2021). Tanzu accelerates applications delivery, manages cloud services, combines development and operations, and speeds K8s adoption (Bowen, 2021). The Army Futures Command is partnering soldiers with VMware Tanzu lab specialists to train soldiers to build cloud-based applications and deploy them anywhere.

Black Pearl has developed its own PaaS offering called Lighthouse (Black Pearl, n.d.). It will serve as an observability, monitoring, logging and alerting platform similar to Big Bang. Like Platform One's Party Bus, Party Barge offers Black Pearl managed security solutions. In 2021 Black Pearl became established under the Navy's program office for digital services and continues to push for accreditation of its tools.

While other DevSecOps offerings run on an Amazon AWS or Microsoft Azure backbone, these four DOD DevSecOps offerings were analyzed by the author for usability with current submarine requirements. Unfortunately, none could be used on submarine hardware and software baselines without significant modification, for both technical and administrative reasons. ACS running OpenShift requires too many resources to run submarine containerized applications on CANES, although it supports Red Hat Enterprise Linux operating systems (NIWC, 2020). During PaaS analysis, Black Pearl, which was initially intended to mirror Platform One, was not far enough along in accreditation for

Lighthouse and was using the Platform One technical stack and baseline. No reciprocity agreement exists between the Army and Navy to use the capabilities of VMWARE Tanzu and that disqualified it, although, later it was discovered that the Marine Coders, a group of innovators in the Marine Corps, had used VMware Tanzu as a DevSecOps offering for its innovation challenge in March 2021. Finally, a reciprocity agreement between the Air Force and Navy was made while Black Pearl was expanding its accreditations and pushing for adoption. This agreement helped Navy teams like Project Blue use the Platform One DevSecOps services and GitOps pipelines to develop their first mission applications and gain expertise with the technical details.

Ultimately, Platform One's Big Bang appeared as the best solution for submarine integration for our project based on its cybersecurity features, intended use within Black Pearl, vendor-independent approach, relationship to Iron Bank, DISA accreditation, open-source nature, small resource requirements in the submarine CANES environment, and success in production. Also, Big Bang core components could be immediately used for Project Blue's submarine-focused PaaS to run its containerized applications. The Project Blue platform will be covered more in Chapter IV.

THIS PAGE INTENTIONALLY LEFT BLANK

III. METHOD

This chapter surveys the context in which the air-gap software delivery tool ZARF was developed, and the need for a submarine specific PaaS to run containerized software on submarines. It describes the software development approach and cybersecurity assumptions made.

A. PROBLEM CONTEXT

In December 2018, the CNO announced an intent to adopt the framework of the “Compile to Combat in 24 Hours” (C2C24) program (CNO, 2018). It emphasized software modernization and interoperability, as well as identified artificial intelligence and machine learning as key focus areas for improving warfighter lethality and readiness. It looked to shorten risk-management framework accreditation, improve cybersecurity monitoring, and shorten the delivery cycle to under 24 hours for Navy warships with a common software life cycle and data environment. It encouraged feedback and identifying barriers for successful adoption and migration to the C2C24 framework.

Previously, NIWC had run a pilot program with several Navy units including the USS Essex (LHD-2) to test C2C24 (CHIPS Magazine, 2018). It used data standardization, shared infrastructure, automation for functional and cybersecurity controls testing, and cloud services, all fundamentals for a DevSecOps pipeline in an operational environment. The pilot program used the CANES infrastructure, the standard computing environment for command, control, intelligence, and logistics services on Navy ships (Space and Naval Warfare Systems Command, 2011).

In December 2020, the CNO announced the availability of the newly approved Rapid Assess and Incorporate Software Engineering (RAISE) Framework for Navy DevSecOps environments and for CANES (CNO, 2020). RAISE further enhanced Navy software accreditation timelines using automation, cyber verification tools, and a Cybersecurity Tech Authority providing certified DevSecOps pipelines (NIWC, 2018). The memo mandated, beginning January 2021, that all programs with new software or upgrades for the surface-ship version of CANES running ACS would use the RAISE

framework. The Navy's first Application Arsenal (applications for Navy ship systems) was later released in August 2021 (Gamboa, 2021). For ACS, CANES, RAISE, and Navy DevSecOps focused teams, NAVWAR supports a Navy DevSecOps pipeline as part of the Project Overmatch and JADC2 initiatives. While much of this was created for the surface Navy, it can also help the submarine force.

B. SOLUTION ASSUMPTIONS

Information from NAVWAR stakeholders helped develop containerized cloud-based software for submarines. CANES configurations used for submarines and ACS documentation provided the submarine hardware and software requirements for a submarine PaaS (PMW-160, 2019; NIWC, 2020). An environment of Red Hat Enterprise Linux versions 7 and version 8 operating systems was chosen for integration tests.

Following review of the documentation, interviews explained what would be needed to overcome the compatibility challenges of ACS with the submarine CANES baseline. It was determined that processor, memory, and storage requirements of ACS were too large, and thus incompatible with the CANES infrastructure onboard submarines (NIWC, 2020; PMW-160, 2020). An alternate DevSecOps PaaS solution was needed. A few interviewees were exploring the U.S. Air Force solution called Big Bang, which sought to achieve lower life cycle costs for DevSecOps in staging and production environments. With budget issues affecting many DOD programs, it was determined by the author and the Project Blue team that the submarine solution should be low-risk, reusable across the DOD, and scale across the submarine platform for efficient software updating during a refit. Furthermore, the reciprocity agreement between the Air Force and Navy Chief Information Officers could reduce authorization and operation timelines (Department of the Navy Chief Information Officer, 2021).

C. SOLUTION DEVELOPMENT

In 2020, the submarine-focused Project Blue team delivered the Beast Core hardened container image to the Iron Bank repository on Platform One. Having passed the certification needed to be in the Iron Bank and having received DISA accreditation for DOD use, a feasibility study to run containers in the submarine environment was next. The

study gathered requirements from the DOD DevSecOps initiatives including Project Overmatch (Navy), Project Convergence (Army,) and ABMS (Air Force) to promote joint software collaboration and avoid redundancy. Performance requirements for the PaaS options were compared to the submarine CANES deployment environment.

Core elements and the open-source concept of the Project Blue PaaS came from Platform One's Big Bang. As a DISA-accredited open-source DOD-owned DevSecOps solution, Big Bang offered a mature starting point for integration into the Navy's Linux operating systems. Figure 1 shows the core components in Big Bang (Platform One Big Bang, n.d.).

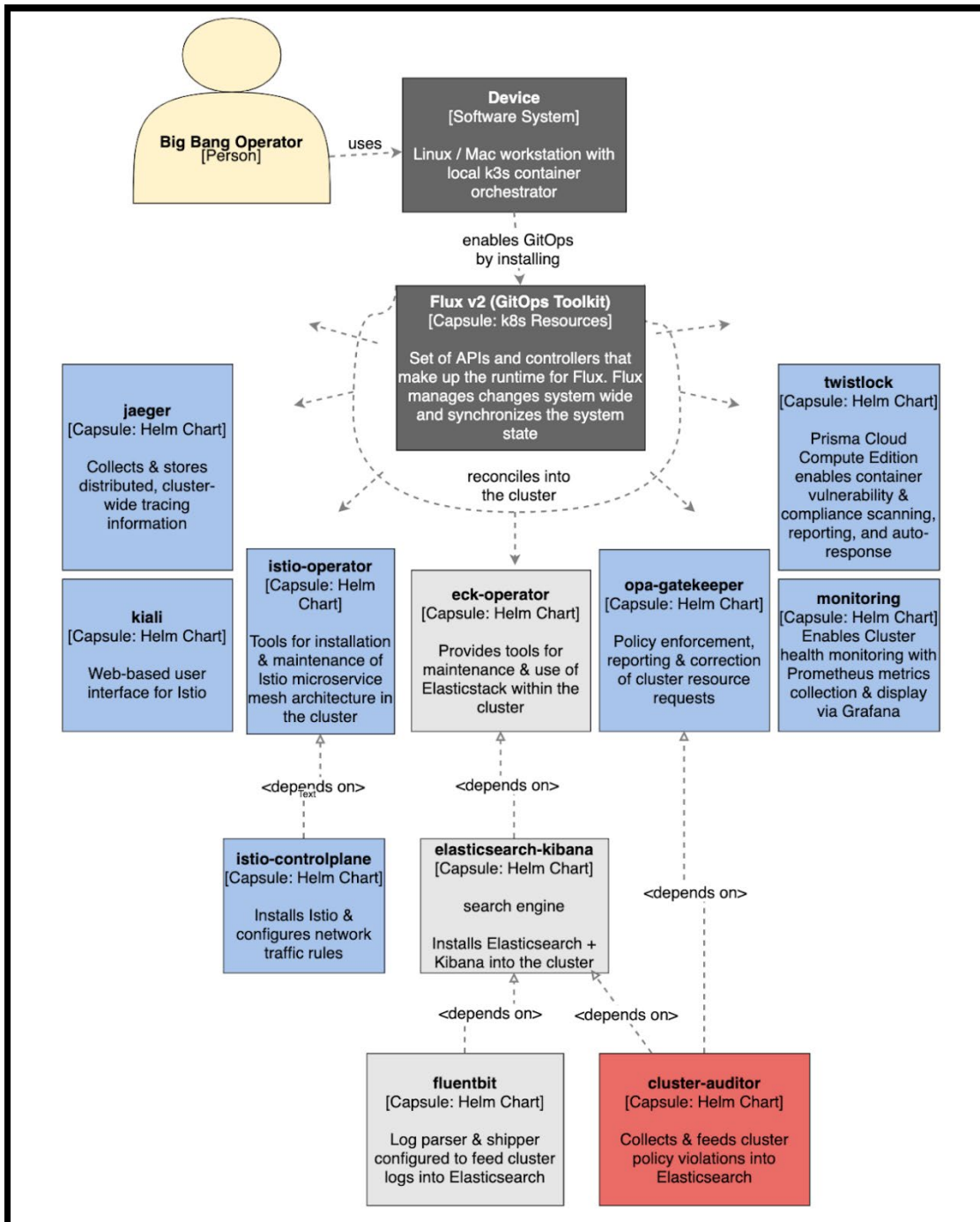


Figure 1. Core elements for Big Bang. Adapted from Platform One Big Bang.

A PaaS for submarines must consider the environment in port, at sea, and for the sailors using the tools. The service should fit in a submarine implementation of CANES while allowing room for other applications. This is important because the Beast Core container requires significant graphics data. The resource requirements for processor, memory and storage were determined from technical specifications provided by the Naval Undersea Warfare Center-Newport CANES test facility and are shown in Chapter VI.

For submarines, isolated (air-gap) environments are needed to avoid exposing potential vulnerabilities until patching is possible. Information systems on submarines are also primarily classified networks, whereas shore support activities, such as the SRSs or TRFs use Unclassified Naval Nuclear Propulsion Information (UNNPI) data. Network segregation requirements and lack of infrastructure further complicate network and data synchronization and hinder maintenance collaboration. Given these limitations, consensus was reached within Project Blue that air-gap software-delivery efficiencies should be pursued for submarines. Finding efficiencies in the air-gap transfer for submarines could apply to other air-gap systems and help with data synchronization needs between the submarine and refit facility. A declarative approach (a combination of GitOps, CaC and IaC) for software delivery was selected to create a repeatable delivery method with as much automation as possible. This could lower the barrier to delivery and provide a scalable solution for long-term sustainability across all CANES submarine platforms (Smith et al., 2021).

A submerged submarine is a weapon system operating at a low bandwidth. Since outside resources and technical guidance are then unavailable, troubleshooting, deploying, or restarting the system must be designed for the skills of the submariners. Leading-edge technologies may be difficult to control. User interfaces, displays, code repositories, threat indicators, system analysis, and recovery methods must be sufficiently intuitive that external communications are not required for troubleshooting.

For end-to-end deployment, the GitOps environment and DevSecOps CI/CD pipelines hosted by Platform One can build, secure, and test containerized, cloud-based applications. This was used for the development of the Project Blue applications and ZARF installation tool. Cloud-based containers can be stored, continuously monitored for

vulnerabilities, and matched to configuration updates in the Iron Bank. ZARF was built using the platform independent GO programming language (GO, n.d.) and provides a declarative approach to software delivery (Github, n.d.). It enabled deploying Big Bang core elements, Iron Bank images, Project Blue's PaaS, the Beast Core mission application, and its digital-twin data to submarine CANES. Hash verification in ZARF ensures file integrity during air-gap transfer. For installation, ZARF provides one-time-use credentials and transport-layer security (TLS) using certificates to do automated network instantiation, configuration, and troubleshooting. Testing was done on an unclassified RedHat Linux 7 CANES virtual machine supplied by NUWC. Interoperability tests on selected operating systems proved portability of the ZARF method and the services using Virtual Box and Vagrant as a sandbox environment.

Core elements of Big Bang keep consistency with DISA accreditation and maintain compliance with DOD and NIST DevSecOps policy. The continuous monitoring and logging tools of Promtail, Grafana, and Loki provided lightweight applications for observability. The PLG stack replaced the heavier Elastic, Fluentd, and Kibana EFK stack used by Big Bang, but display the same representations of behaviors that could indicate possible compromise or abnormal system behavior. Gitea provided a small Git download environment to maintain an auditable changelog and repository for utility-cluster data. K3s was used as an open-source CNCF container orchestrator for deploying K8s to edge devices. The Docker registry and Containerd were taken from the Iron Bank. For K3s pod and node monitoring and manipulation, K9s provided a small open-source user interface for K8s to allow a user to interact with the node architecture. K9s provides an operator-friendly substitute interface for the industry standard Kubectl. With ZARF, the Beast Core mission application and interfaces for the Project Blue PaaS were bundled and chosen to minimize the resource requirements.

IV. APPROACH

This chapter covers the key components of Project Blue’s PaaS called CocoWow, which provides a lightweight open-source solution for running containerized applications on submarines and achieving DevSecOps cybersecurity goals. Specifics are provided for the software delivery tool ZARF, which helps streamline air-gap delivery and create repeatable installation. Details for the changes to Big Bang to run on submarine hardware and software baselines are also discussed in this chapter. This work’s main contribution to submarine DevSecOps is ZARF appliance mode, which replaces the Big Bang Elasticsearch, Fluentd, and Kibana (EFK) services with the Promtail, Loki, and Grafana (PLG) services. Using the PLG services for the Project Blue PaaS keeps cybersecurity and observability goals for submarines.

A. PROJECT BLUE’S PLATFORM-AS-A-SERVICE

CocoWow (Figure 2 and Figure 3) provides a DOD DevSecOps-compliant capability for submarines. It has the necessary services to meet DevSecOps observability goals for system performance monitoring and cybersecurity management. This PaaS provides a lightweight container orchestrator and virtual security-operations center (vSOC) for running K8s clusters on submarine CANES Linux operating systems following the PMW-160 CANES Roadmap. Alerts and indications when using CocoWow’s services can reveal when a system requires attention or has abnormal behavior.



Figure 2. Official Project Blue PaaS logo

The tools in CocoWow were based on best practices for monitoring and observability as defined by Google Cloud architecture guidelines (Google Cloud, n.d. a; Google Cloud n.d. b), the DOD DevSecOps Playbook and the DevOps Research and Assessment (DORA) studies. The Google Cloud guidelines specify these:

- Transparency of the overall health of systems functioning, resource availability, as well as communication of system outages
- Visibility of leading indicators of outages, service degradation, bugs, and unauthorized activity
- Monitoring for key business and systems metrics and long-term trends for capacity planning and business purposes
- Access to tooling to help troubleshoot and debug systems in production
- Access to tooling and data to help trace, understand, and diagnose infrastructure problems in the production environment, including interactions between services

- Identifying unknown unknowns for risk and contingency planning, as well as the ability to expose unexpected side effects of changes or added functions.

The Project Blue team specially considered the tactical environment of submarines in the design for CocoWow. Using the PaaS, submariners must observe and troubleshoot behaviors in a disconnected or contested environment. Surfacing or coming to periscope depth may not be an option if the system requires outside technical support. A balance between lightweight capability to fit in the constrained CANES environment and observability of behaviors within the system had to be maintained.

Useability, intuitiveness, and ability to manage routine operations were prioritized for user presentation in CocoWow. The submarine sailor lacks the time and training that a software developer would have and must rely on basic training and continuing education programs to troubleshoot in an isolated environment. For newly developed systems, senior leadership with years of experience will be unavailable, meaning a more objective, systematic approach to troubleshooting and system operations is required. New systems may do things not seen before. CocoWow tries to ensure a submariner has the tools needed to operate its technology and benefit from its capabilities.

CocoWow was derived from Big Bang, a DevSecOps PaaS deployable software factory and container orchestrator. The biggest change for CocoWow from Big Bang occurred in the separation from the vSOC EFK services, which were replaced with a smaller PLG stack:

- Promtail - used for application monitoring and log shipping to Loki (Grafana Labs Promtail, n.d.)
- Loki- infrastructure log aggregation and labeling (Grafana Labs Loki, n.d.)
- Grafana- computer resource metrics and observability interface (Grafana Labs, n.d.)

The PLG stack provided similar observation and functional services as the EFK stack but requires fewer computing resources. Also, lightweight user-friendly graphical

interfaces and syntax-friendly options were included. K9s was included as a K8s architecture, services, and applications visualization tool to substitute for the heavy-syntax Kubectl standard (Derailed, n.d.). Gitea was included as a small self-hosted Git service for the code repository when a Kubernetes utility cluster will be deployed on a larger scale (Gitea, n.d.). Gitea can also store the data associated with large applications, such as digital-twin data, and provides an alternate data-loading method for the running K8s clusters. K3s was selected as a certified CNCF K8s service, optimized for “unattended, resource-constrained, remote locations or inside IoT [Internet of Things] appliances” (Rancher Labs, n.d.; Containerd; 2021).

Table 1 and Figure 3 summarize the services used with deploying the Beast Core container on CANES using the Project Blue PaaS.

Table 1. Summary of Project Blue PaaS and Beast Core mission services.

Project Blue’s CocoWow PaaS and mission application	
Services	Functions
K9s	Kubernetes (K8s) graphical interface, command-line interface, and metrics-visualization tool. Substitute for the K8s industry standard Kubectl (K9s, n.d.)
Gitea	Changelog and Git source for utility-cluster data for larger-scale applications (Gitea, n.d.)
Promtail	Event logging, monitoring, and alerting service (Grafana Labs Promtail, n.d.)
Loki	Infrastructure log aggregation and labeling service (Grafana Labs Loki; n.d.)
Grafana	System performance dashboard for the metrics and monitoring platform (Grafana Labs; n.d.)
Kiwigrid/k8s-sidecar	Kubernetes pod monitor for auditing clusters (Platform One Big Bang, n.d.)
K3s	Edge-optimized container orchestrator (Rancher Labs, n.d.)
Twistlock	Container runtime security service (Platform One Big Bang, n.d.)
Beast Core	Container for three-dimensional model and technical systems documentation viewer (Beast Code, n.d.)

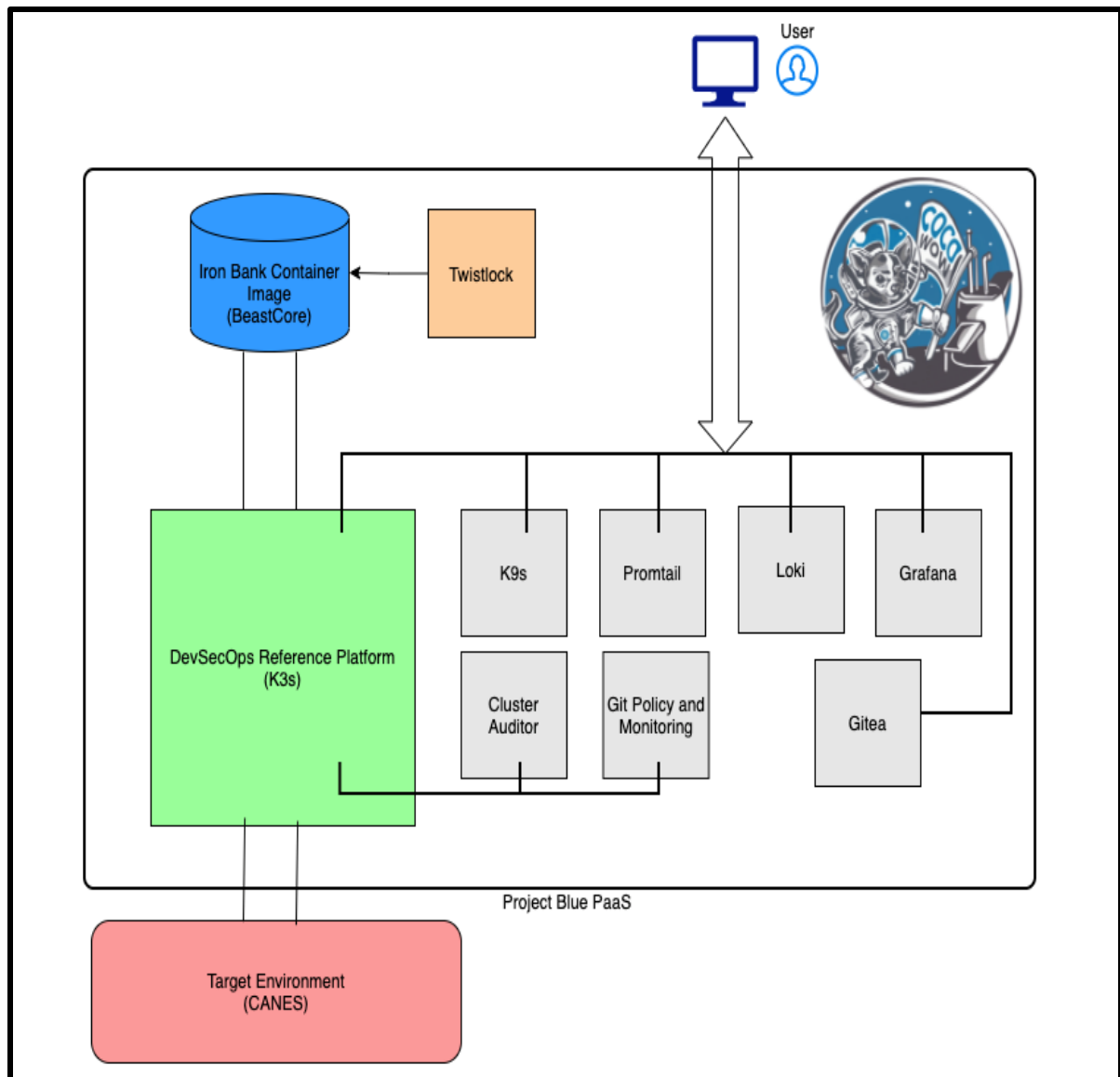


Figure 3. High-level architectural view of the services chosen for deployment to the CANES infrastructure

B. ZARF AIR-GAP DELIVERY TOOL

Installing updates can be laborious with older technology on submarines. ZARF streamlines software delivery over the air gap by using a declarative delivery state for scalability, repeatability, and ease of installation. This approach uses GitOps, IaC and CaC to define the desired installation goal. This declarative state means the software developer defines the system configuration information before installation, automating and simplifying the process (Smith et al., 2021). The original integration tests for ZARF were limited to submarine use but could extend to delivery challenges for other DOD air-gap systems. For submarines, two key considerations went into the creation of ZARF: (1) hosting Unclassified Naval Nuclear Propulsion Data (UNNPI) in the cloud, and (2) exposing legacy submarine technology to the Internet, expanding the cyberattack surface of the weapon system.

To deploy DevSecOps services and Project Blue’s submarine applications, ZARF must manage the bottleneck and sustainability issues a manual transfer of software between air-gapped systems creates. This addresses baseline drift (i.e., different ships on different versions of the same CANES baseline) which complicates installation. Installation overhead or technical debt gained from the current procedural software delivery approach, as observed in both the testing and operational CANES environments for submarines, prevents fast and scalable software delivery. Baseline drift poses a significant issue to continuous integration and continuous delivery because deviations from a platform-common baseline may prevent consistent, repeatable, and fast software changes across the fleet (Smith et al., 2021). With ZARF, a scalable and repeatable method is available using the declarative state for different versions of the CANES baseline.

ZARF (Figure 4) has two phases in installation. Phase 1 ensures proper configuration and instantiation of the PaaS while Phase 2 deploys the application and loads the data into the K8s architecture. Details are given in Appendix D. This separation was made with troubleshooting in mind. ZARF also has a command to remove its artifacts from the system to allow a restart from a clean state.



Figure 4. Official ZARF logo

Three modes for software delivery to air-gapped systems are possible using ZARF as well as custom modes. For the CANES lab testing discussed in Chapter VI, Appliance mode bundled K3s resources and delivered them to a CANES baseline to run the Beast Core container. Creating appliance mode required changes to Platform One’s Big Bang solution to ensure resources are within the capabilities of the submarine version of CANES. The three modes are now discussed in this chapter with their relationship to Big Bang highlighted. Edge mode and Data-Center mode are standard Big Bang modes optimized using ZARF.

1. Appliance Mode (Single-Use Mission Applications):

Appliance mode for ZARF brings K8s resources and dependencies to low-resource single-application environments. In this mode, and for the test case in Chapter VI, a K8s utility cluster is not required. By contrast, in larger data-center environments, a utility cluster is desirable when system-load management and resource balancing supports many users, services, and applications. An example of a single-use mission application in Appliance mode is the Beast Core container running on a standard CANES RedHat Linux 7 virtual machine using K3s as the container orchestrator and the PLG vSOC. As the only container in the architecture, no other applications compete for resources with Beast Core.

Testing a K8s architecture this way was a good start in the feasibility study for submarine CANES lab testing.

In Appliance mode, ZARF provides a simple way to bundle resources and dependencies of K3s for air-gap transfer. ZARF deploys a basic K3s cluster using Traefik 2 (the K3s traffic load balancer). It also configures transportation-layer-security to deploy Podinfo for secure network communications, a small web application for running microservices in K8s (Prodan, 2021). Podinfo is used by other GitOps projects like Flux and Flagger for end-to-end testing. Flux does data reconciliation across different application interfaces and is used by Big Bang in data-center type deployments as in Figure 1 of Chapter III. Definitions of the Git monitoring and policy deployments for K8s security enable a flexible observability platform. For example, smart-card authentication for accessing a UNNPI cloud may be required in a bring-your-own-device environment but not for a sailor with a username and password on a submarine information system. Git policy and monitoring, and the other elements of Big Bang, are discussed in the documentation provided by Platform One (Platform One Big Bang, n.d.).

Successful combination and deployment of the default tools in ZARF Appliance mode, the PLG stack, and the Beast Core application were demonstrated at the CANES test lab at NUWC-Newport. These results are discussed in Chapter VI. Figure 5 summarized the elements that went into the creation of the Project Blue PaaS. Figure 6 specifies the services used in ZARF Appliance mode. Figure 7 highlights the relationship of ZARF Appliance mode to Big Bang.

$\begin{aligned} \text{Project Blue PaaS} = & \text{PLG observability platform} \\ & + \text{Big Bang core elements} \\ & + \text{Mission Applications and Data} \end{aligned}$

Figure 5. Generalized Project Blue PaaS and ZARF components

Project Blue PaaS = (Promtail + Loki + Grafana)
+ (Cluster-auditor + Git policy and monitoring)
+ Twistlock + Kiwigrid/k8s-sidecar)
+ (Beast Core container and data)

Figure 6. Project Blue PaaS services

Big Bang elements included in appliance mode with source listed:

1. Cluster-auditor
 - repo1.dso.mil/platform-one/big-bang/apps/core/cluster-auditor
2. Git policy and monitoring
 - repo1.dso.mil/platform-one/big-bang/apps/core/policy
 - repo1.dso.mil/platform-one/big-bang/apps/core/monitoring
3. Twistlock
 - repo1.dso.mil/platform-one/big-bang/apps/security-tools/twistlock
4. iwi-grid/k8s-sidecar
 - registry.dso.mil/platformone/bigbang/apps/core/monitoring/kiwigrid/k8s-sidecar

Figure 7. Big Bang elements in ZARF Appliance mode

2. Edge Mode (with Istio Service-Mesh)

In Edge mode, an Istio service “mesh” handles when a system grows from a single mission application to multiple services and containers. Istio manages the microservices running in a K8s based system (Istio, 2021). Istio provides an efficient way to secure, connect, and monitor services within a mesh, and is integrated into Big Bang. Jaeger is a tool to troubleshoot problems in distributed service meshes (Jaeger, 2021). Kiali provides dashboards for operators to view features such as network topology and system health (Kiali, n.d.). The combination of the services in Appliance mode and those in Figure 8 comprise Edge mode. Figure 9 highlights the additional elements of Big Bang include in ZARF Edge mode.

Edge mode = (Appliance mode) + (Istio + Jaeger + Kiali)

Figure 8. ZARF Edge mode services

New Big Bang elements included in ZARF edge mode with source listed:

1. Istio control plane
 - repo1.dso.mil/platform-one/big-bang/apps/core/istio-controlplane
2. Istio-Operator
 - repo1.dso.mil/platform-one/big-bang/apps/core/istio-operator
3. Jaeger
 - repo1.dso.mil/platform-one/big-bang/apps/core/jaeger
4. Kiali
 - repo1.dso.mil/platform-one/big-bang/apps/core/kiali

Figure 9. Additional Big Bang elements in ZARF Edge mode

3. Data-Center Mode

The EFK services for utility-cluster management consists of Elasticsearch, Fluentd, and Kibana. Elasticsearch collects logs monitored by the platform (Elastic, n.d.). Fluentd gathers logs from within the system and feeds them to Elasticsearch. Kibana is a Web interface for Elasticsearch. Notably, EFK is a portable option used by other K8s orchestration platforms such as RedHat OpenShift, the template for the Navy DevSecOps offering (RedHat, n.d.). Adding the EFK stack and its operator service to Edge mode in Figure 10 creates Data-Center mode and a full Big Bang deployment. Figure 11 highlights the EFK services included from Big Bang in ZARF Data-Center mode.

Data-Center mode = (Appliance mode) + (Edge mode) + (EFK)

Figure 10. ZARF Data-Center mode

New Big Bang elements included in ZARF data-center mode with source listed:

1. Eck-operator

- repo1.dso.mil/platform-one/big-bang/apps/core/eck-operator

2. Kibana

- repo1.dso.mil/platform-one/big-bang/apps/core/elasticsearch-kibana.git

3. Fluentd

repo1.dso.mil/platform-one/big-bang/apps/core/fluentbit

-

Figure 11. EFK elements from Big Bang in ZARF Data-Center mode

THIS PAGE INTENTIONALLY LEFT BLANK

V. CYBERSECURITY CONSIDERATIONS

ZARF addresses some technical challenges of deploying secure software in air-gap environments like submarines. Other considerations are discussed here for building security into the software delivery.

A. ESTABLISHING CHAIN OF TRUST IN AIR-GAP DELIVERY

Air-gapped delivery of software reduces the risks of cyberattack vulnerability for a submarine. Connecting a submarine to a DevSecOps pipeline on the Internet is not currently available. A DevSecOps pipeline for submarines can aid software sustainment, cybersecurity, and updates, but the weapon system must remain secure when it connects to new software through air-gap transfer or in a future-connected state. Ransomware is an issue because cyberattacks target industrial control systems, resulting in failures, operations delay, and damage to oil pipelines, dams, and power grids (Butt et al., 2019). A submarine is an industrial control system because its nuclear power plant supplies propulsion, electricity, and supports weapons operations. A cyberattack compromising confidence in the weapon system is a threat to national security, especially for SSBNs, which holds 70% of the U.S. nuclear weapons (Korda, 2021).

Another reason to explore air-gap deployment is the technical limitations of the legacy submarine infrastructure. The Ohio class submarines and their supporting infrastructure for the TRFs were built beginning in the 1970s (Eckstein, 2020). Submarine construction lags the advances in technology between builds and class changes. This introduces challenges for including cloud-based technologies like the cloud-native access point (CNAP), while meeting the stringent Naval nuclear-propulsion information (NNPI) handling requirements (DOD, 2021e).

The chain of trust for an application or service begins with building a software-development team. Team choices, coding practices, and priorities with respect to cybersecurity become observable with new code creation and revisions in a DevSecOps pipeline. An attacker can still exploit many things in air-gap development and delivery. They can directly target a source-code repository (such as GitLab where ZARF is hosted),

use brute force or social-engineering methods to break into a network, or steal data using side channels such as by monitoring wireless connections. Defense-in-depth practices such as multi-factor authentication can help prevent such activities; Platform One uses the Single Sign On multi-factor approach for user access.

In software development, automated static and dynamic code reviews using openly available tools like Snort, Fortify, and Trufflehog help identify possible vulnerabilities as code changes are made. Additional automated security tools and code dependency checks can monitor for poor coding and enforce code-coverage metrics for applications. Insider-threat training, red teaming, penetration testing, continuous monitoring for CVEs using repositories like the NIST CVE databases, using tools like Twistlock for container monitoring, and using cloud-based access points, such as the CNAP as Platform One does to create their software-defined network perimeter and zero-trust architecture are all practices observed during this study that make it difficult for an attacker to exploit Project Blue and ZARF software. These defense-in-depth measures can lower the risk of bugs and vulnerabilities being introduced into source code and help identify integration issues before they make it into a weapons system.

As the DOD continues to focus on a future with artificial intelligence, data analytics, and default connectivity, focusing on cybersecurity is important. Although achieving perfect security and complete confidence a system will not be vulnerable to a cyber threat is impossible, it is possible to make it much harder for an attacker to succeed. Air-gap transfer methods are not enough. Raising the bar will require DOD investment in capability and competency building with a focus on human factors. Added complexity introduced by the human element can be managed by presenting principles of DevSecOps to enable a culture focused on creating cybersecure and capable systems on both sides of the air gap.

B. ZARF AIR-GAP TRANSFERS

Figure 12 summarizes ZARF's process beginning with the bundling of resources after software development. This process maintains the integrity of the data and the software transfer. Items in grey indicate key points.

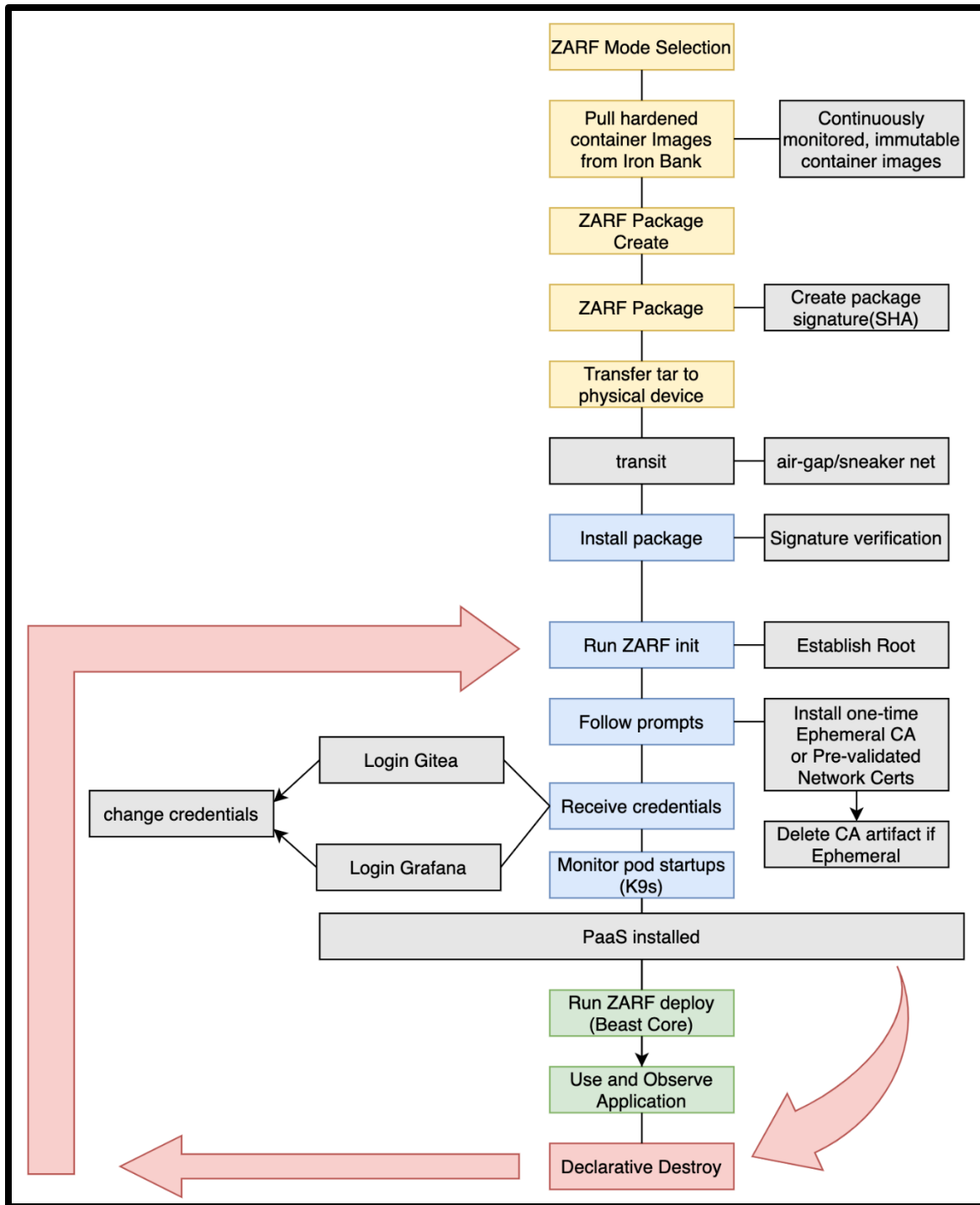


Figure 12. Process of a ZARF air-gap transfer

Some details on the figure:

Continuously monitored, immutable container images: ZARF pulls hardened container images from the Iron Bank. Iron Bank continuously scans containers for configuration updates, known vulnerabilities, and known exploits using backend sources like NIST. This prevents known vulnerabilities getting onto the weapons system. It does not remove the chance of a new vulnerability from getting on to the system, but significantly reduces the cyberattack surface.

Create package Signature: ZARF runs the Secure Hashing Algorithm (SHA) on the files bundled in the transferred file as well as on the entire file. This can be compared to the posted hash values for these files to later show the data is unmodified.

Air-gap/sneaker net: Air-gap deployment keeps the submarine network isolated from the Internet and lowers the risk of an unpatched vulnerability being discovered.

Signature verification: A ZARF file has a unique signature based on its contents due to the properties of a hash function. The signature on the connected environment should match the signature in the air-gap environment, and this should be checked both in Iron Bank and before installation.

Establish Root: Privileges with files should be guided by the Principle of Least Privilege. Only system administrators should have root access. Changes to the network and server should only be done by an authorized administrator.

Install a one-time ephemeral Certificate Authority (CA) or pre-validated network certificates: ZARF can create a single-use certificate for a faster alternative initialization of the transport-layer security required to communicate with Kubernetes.

Logins to Grafana and Gitea: ZARF creates single-use passwords to access the application program interfaces to monitor for performance and compromise.

PaaS Installed: ZARF installs an observability platform for system metrics monitoring, log aggregation, security alerting, and troubleshooting. This contains the virtual security operations center (vSOC) required by the DevSecOps Reference Design Architecture, and reports behaviors that may indicate a compromise.

VI. RESULTS

This chapter documents the conditions, products, and the results of the feasibility testing of the Project Blue PaaS using the ZARF air-gap delivery tool. The submarine CANES baseline selected was HW1.2SW3 with resources listed in Table 2. An overview of the ZARF interaction is in Appendix D.

Table 2. Resources available for submarine CANES

HW1.2 SW 3 Resources	Total available
Virtual processors	56 virtual CPUs
Memory (GB)	256 Gigabytes
Disk storage (TB)	8 Terabytes

To handle the limitations of Table 2, the Big Bang PaaS was adapted to create CocoWow, a submarine PaaS to run cloud-based containers in a lightweight configuration. The resulting services, memory, and processor allocations are shown in Table 3.

Table 3. CocoWow resource requirements

Project Blue's CocoWow PaaS and Mission Application		
Service	Memory (MB/GB)	Processors (CPUs)
K9s (*)	Steady State: 50MB RAM Max: 739 MB RAM	Steady State: 0.5 CPUs Max: 1 CPUs
Gitea	Steady State: 512 MB RAM Max: 2 GB RAM	Steady State: 0.1 CPUs Max: 1 CPUs
Promtail	Steady State: 256 MB RAM Max: 3 GB RAM	Steady State: 0.25 CPUs Max: 2 CPUs
Loki	Steady State: 128 MB RAM Max: 2 GB RAM	Steady State: 0.1 CPUs Max: 1 CPUs
Grafana	Steady State: 128 MB RAM Max: 2 GB RAM	Steady State: 0.1 CPUs Max: 1 CPUs
K3s (**)	Steady State: 512MB RAM Max: 1GB RAM	Steady State: 0.250 CPUs Max:
Docker-registry	Steady State: 512 MB RAM Max: 2 GB RAM	Steady State: 0.1 CPUs Max: 1 CPUs
Twistlock	Steady State: 256MB Max: 256MB	Steady State: 0.1 CPUs Max: 1 CPUs
Cluster auditor	Steady State: 2GB Max: 2 GB	Steady State: 0.1 CPUs Max: 0.5 CPUs
Beast Core container	Steady State: 512 MB Max: 2GB	Steady State: 0.125 CPUs Max: 1 CPU
Beast Core data	25GB storage	

* Command line tool. Not always running and negligible when unused.

**Implementation details for max K3s resources is atypical. K3s scales up based on system load.

A. TESTING

1. CANES Lab

The feasibility study concluded with testing at the Naval Undersea Warfare Center in Newport, Rhode Island. Table 4 shows the conservative build specifications based on resource requirements of both the Big Bang and the Project Blue PaaS, as well as the data requirements for running the Beast Core container.

Table 4. Conservative virtual machine pre-build specifications for CANES lab testing

	Beast Core data	CocoWow (PaaS + Beast Core container)	Total	% CANES
CPUs	-	12	12	21%
Storage	25 GB	5GB	30GB	0.4%
RAM	-	32	32GB	12.5%

To test intuitiveness, the container installation and PaaS tests were done in two parts. Configuration for the CANES baseline was done by the NUWC Application Integration team and Project Blue software developers. They simulated the information-technology and software teams who would be brought in during maintenance for an application update or system overhaul on a submarine. The author then acted as the stand-in submariner due to his warfare qualifications. Unaided, the author used the tools provided in ZARF to deploy CocoWow and use Grafana, Loki, Gitea, K9s, and Beast Core. The author was involved in the creation of ZARF, so he may have had some bias. Nonetheless, this tested whether typical personnel on a submarine would have a satisfactory level of knowledge to do these tasks on a ship either in port or at sea.

SUCCESS for the CANES lab integration tests using ZARF met the following criteria:

- Installation of the K3s Kubernetes architecture
- Login and observe Grafana
- Login and observe Loki
- Verify log capture for Promtail
- Login and observe Gitea
- Observe system functions, services, metrics with K9s
- Observe Beast Core function

Table 5 summarizes the tests. ZARF integration tests were completed outside the lab on a RedHat 7 and 8 operating system in preparation for the lab environment. The NUWC team provided the RedHat 7.4 and 7.6 virtual machine images on testing Day One. The Nosis program office provided the installation team with a Nosis RedHat 8 virtual machine image on testing Day Two. The Ubuntu operating system and Doom container were brought by the Project Blue Team.

Table 5. Submarine platform integration tests

Submarine compatibility	
HW1.2SW3	Integration tests
Ohio RedHat Linux 7.4	<u>SUCCESS</u>
Ohio RedHat Linux 7.6	<u>SUCCESS</u>
Virginia RedHat Linux 8	<u>SUCCESS</u>
Ubuntu Linux DOOM	<u>SUCCESS</u>

2. Sandbox Testing

Additional testing of ZARF was done on a variety of operating systems using a Virtual Box and Vagrant machine sandbox environment before CANES lab testing. This demonstrated the portability of ZARF to other environments.

PASSED for operating systems integration testing using ZARF shown in Table 6 meant successfully completing the following criteria:

- Installation of the K3s Kubernetes architecture
- Login and observe Grafana
- Login and observe Loki
- Verify Log Capture for Promtail
- Login and observe Gitea
- Observe system functions and view metrics with K9s

Table 6. ZARF sandbox testing

ZARF operating systems compatibility	
RedHat Linux 7	<u>PASSED</u>
RedHat Linux 8	<u>PASSED</u>
CentOS 7	<u>PASSED</u>
CentOS 8	<u>PASSED</u>
Ubuntu Linux	<u>PASSED</u>
Debian Linux	<u>PASSED</u>
Rocky Linux	<u>PASSED</u>

THIS PAGE INTENTIONALLY LEFT BLANK

VII. CONCLUSION

The major achievements of this research are (1) demonstration of technology and a trusted method of delivery for containerized software in a connectivity-constrained and resource-constrained environment using GitOps; (2) demonstration of the feasibility of a software-delivery model (Smith et. al. 2021); (3) prototyping of logistics software for submarines and maintenance-support facilities; (4) contributing to knowledge of how to apply DevSecOps to submarines.

Successful CocoWow installation using ZARF on the CANES servers for the HW1.2SW3 baseline demonstrated a way to deploy cloud-based containerized applications for submarines, edge systems, and air-gap environments while satisfying cybersecurity objectives. This broadens the options available for integration with JADC2 and Team Submarine. It is consistent with the Digital Transformation goals of the DOD and looks to further the CNO's goals for fleet readiness. Our work demonstrated that using Big Bang and CocoWow as container orchestrators enables containers to be delivered to a submarine baseline in a declarative state using ZARF. Pairing this research with the quantitative framework for cost comparisons between declarative and imperative states will allow the Navy to extend this research (Smith et al., 2021).

A. CONCLUSIONS FROM THE TESTING

The successful integration tests in the CANES test facility at NUWC-Newport confirmed adequacy of the DevSecOps methods. However, the three-dimensional modeling graphics only partially rendered on the testing laptop due to the graphics limitations and older application versions on the device. The technical-documentation viewer for the ship systems within Beast Core was available, and the services for CocoWow worked as designed, so the approach and deliverables for the study were mostly a success. A containerized version of the video game DOOM was deployed using ZARF in Appliance mode to show that other legacy Microsoft software on submarines could be containerized.

The testing environments and legacy infrastructure that the Navy currently operates have significant cybersecurity issues. An air-gap environment is not ideal for aging

technology. Cyber compromise of a system across the air gap can happen, and without up-to-date and patched systems, compromise by an attacker is easier. Using years-old technology because of the slowness of administrative and bureaucratic practices is unacceptable in today's cyber landscape. Adopting and integrating new technology in submarine environments remains a challenge given the constraints of the onboard submarine infrastructure and the rapid pace of technology advancements beyond the procurement system.

In the CANES lab testing, partition issues delayed the configuration and installation of the Beast Core container. This unforeseen obstacle revealed the data-management requirements for Beast Core, although the Beast Core data was only a small sample of what a complete Ohio-class submarine would have been. This process must be improved to ensure the footprint of the digital twin does not overrun the system capacity. New workstation hardware will likely be required. Also, in a virtual machine environment, abstraction using graphical interfaces makes building and balancing resources easier. However, transferring gigabytes of digital-twin data requires time, which makes troubleshooting difficult and significantly slows installation.

B. RECOMMENDATIONS

Technical challenges met were largely due to existing infrastructure and practices.

- A declarative approach to software delivery is necessary to meet Navy warfighting objectives. A GitOps environment with a single declarative baseline is critical for scalable DevSecOps solutions and should be adopted by the submarine facilities. This has been achieved in varying degrees of success by the Project Overmatch Collaborative Software Armory, the Platform One Party Bus, Black Pearl Party Barge, and the pipelines at Navy PEO-Integrated Warfare System called The Forge. Within these pipelines, teams can access representative environments which aid interoperability and minimize breaking changes, which can be identified and tested before being delivered to the platform. A representative environment is readily available with the CNAP.

- External hard drive size has decreased while storage capacity has grown significantly. Edge storage devices like Amazon Snowball could help applications like digital twins on a submarine where space is limited.
- VMware Vsphere cloud management has issues running Kubernetes. PuTTY (a secure-shell and client-connection application) was a substitute to avoid creating artifacts that would break the Kubernetes architecture. VMware should try to handle this increasingly popular service.
- Classification guidance should better address cloud-based technology and modern accreditation frameworks. Default classification levels of Secret or higher for submarines challenge the adoption of new technologies and complicate software delivery. Much maintenance information and data useful for analytics is unclassified and could be shared across organizations or networks. An accredited cloud repository for UNNPI data at the Trident Refit Facilities and onboard the submarines could support analytics, artificial-intelligence, and machine-learning applications. Studying and applying this model for the Columbia ballistic-missile submarine class should be explored.

C. COMMITMENT TO DEVSECOPS

Cultural challenges were observed during the study. The USG and DOD must commit to DevSecOps to continually adapt and deliver capability fast enough to keep warfighters from losing a competitive edge or becoming disadvantaged. Leadership must emphasize that applying lessons learned and finding success across organizations (versus competing with them) is necessary to rapidly modernize information technology and maximize availability and readiness by reducing maintenance inefficiencies, including updating software. Program managers (with direct support from their product support managers) under Title 10 U.S. Code § 2337 have single-point accountability for sustainment objectives, to include continuous information-technology support with considerations for DevSecOps (DOD 2019b). A principal duty includes evaluating and exploiting opportunities across programs to improve use of industry and DOD resources. Despite this mandate, some Navy stakeholders

interviewed expressed frustration about their leadership's resistance to adopt Big Bang due to its U.S. Air Force source.

The processes and culture within the current software environment are insufficient to rapidly modernize technology and improve cyber posture. This must change in the interest of national security (DOD, 2018a). The introduction of new ideas and mandates naturally bring resistance and discomfort, as a shift from the status quo may not be well understood, or seen as necessary (Cameron & Green, 2009). Reciprocity agreements are an example. When they are in place, trusting in the hard work and competency of those involved versus reviewing and validating their legitimacy should occur. The latter does the same work twice, displays an inherent lack of trust, and wastes valuable time, yet DOD components repeatedly delay reciprocity (Barnett, 2021). Reciprocity should not excuse due diligence by a program but should not hurt the efficiency reciprocity will create.

The Navy should look to certify the DevSecOps work being done by each service to hasten joint solutions and create a faster path to delivery when pathfinder teams identify breakthrough solutions. This may not be most efficiently achievable through eMASS. The eMASS system is not uniformly implemented by all services (Department of Defense Education Activity, 2019) and needs a significant software reorganization. It is burdensome, not intuitive, and its training provides inadequate information for Navy accreditation.

Regardless of whether reciprocity is part of the overall strategy, cybersecurity subject-matter experts should be included in development teams so that architectural and software designs meet basic cybersecurity needs and co-developing the accreditation strategy happens long before pushing code to pipelines for production. Poorly designed systems are just as much of a threat as poorly coded systems. Acquisition and architectural problems will continue to be relevant as the next generation of ships are developed and delivered.

Cybersecurity training should be continued fleet-wide but also includes the idea that perfect security does not exist. It is important to emphasize that cybersecurity is everyone's responsibility, and the job of a cybersecurity professional is not to be perfect, but to manage the risk that comes with connecting systems. There will always be risk and inevitable failures as the DOD looks to modernize with its new strategy. Creating a culture that understands

cyberspace risk, balances them against connectivity benefits, and supplies training and tolerance for failure as systems are modernized will enable a more agile and cybersecure environment.

D. SOFTWARE IS NEVER DONE

An agile mindset that “software is never done” must resonate in acquisition and operations leadership. Software sustainment should be regarded as highly as physical sustainment. Modern software engineering best practices revolves around agile development and continuous improvement. It is not waterfall-based with phases that are completed and never revisited. Software must get into production quickly, although it may fail, to understand critical requirements and progress in a dynamically changing environment. The prototyping of The Bridger Project, the Master Job File, and the Beast Core products occurred over six months. Convincing the appropriate channels to pilot these tools took twice as long as their development and is ongoing.

The Project Blue feasibility study suggests that predefined requirements are unnecessary for innovative cultural changes. No requirement exists for a submarine PaaS, although it is desired. Some of the best innovations, such as the Sidewinder Missile, have come about organically and through an agile approach.

E. INTEROPERABILITY ACROSS COMPONENTS

The JADC2 initiative seeks to integrate sensors from the services’ representative programs: Project Overmatch, ABMS, and Project Convergence. Collaboration will be critical for achieving the vision of the DevSecOps policy designs and architecture interoperability. Outside of these programs, countless teams and innovators in the DOD ecosystem operate in data-siloed environments, looking to build enterprise synergy, and organically build solutions to meet mission needs inadequately addressed by the programs of record. These innovative teams likely possess the desire, agility, talent, and speed to combine the critical pieces to the wickedly complex puzzle that is JADC2 and achieve DOD objectives in the current cyber-driven software-defined conflict. Innovative solutions can occur outside the traditional top-heavy, formal, command-and-control DOD structure. Relying on a single solution stemming from a program of record (with ownership of the requirement), is an

insufficient strategy to develop and implement the needed capabilities and modernizations to remain strategically relevant. Helping small, organic, discovered teams bring solutions across the bureaucratic “Valley of Death” will be critical to organizational success.

Although the Navy has the Application Arsenal, it is tailored to ACS. The Navy could benefit from studying the Iron Bank as a DISA-approved source for hosting, scanning, and monitoring containers for threats and vulnerabilities. Under reciprocity with Black Pearl, Navy teams like Project Blue are already using this service. This would avoid rework in accreditation systems like eMASS, the Navy’s commonly used software accreditation pathway. Similarly, improvement in the documentation of Big Bang would help the broad stakeholder community looking to use it. Communicating to the other services about development, security, and operations is critical to building new relationships that support joint initiatives. The lack of formal means, ability, or bandwidth to produce it when asked, is a problem. However, the Navy would benefit from accrediting and authorizing Big Bang for broad use, as it would enable teams to use a vendor-independent platform. This would democratize the environment for innovation and experimentation, allowing pursuits of DevSecOps goals to begin without having to buy services in a budget-constrained environment. It will also enable further joint-service collaboration.

F. ONE TEAM, ONE FIGHT

Throughout this study, teams interviewed expressed concerns with Navy software systems being slow, unresponsive, and bureaucratically managed. However, while they acknowledged improvements had been made, there remained disinterest in adopting solutions that are not their own and warned about Platform One. Many excellent Air Force, Navy, Space Force, Army and Marine Corps teams gave objective feedback and offered guidance to the author that was critical to the success of this thesis.

In the author’s opinion, the many innovation teams interviewed looking to implement DevSecOps in their programs believed in ideas central to the culture of DevSecOps. These beliefs were held despite the timelines given to them to move their products into production. The DevSecOps ideals that resonated within these teams included (1) industry best practices of agile-software development enable flexible and secure product management; (2)

continuous integration and continuous delivery can accelerate relevant outcomes for warfighters; and (3) cybersecurity with continuous monitoring must be prioritized. This parallels the sentiments seen in the various articles, policy, and anecdotal experiences for DOD's adoption of DevSecOps and how it translates to success in a technology-driven and software-centric 21st century.

On the current course without synergy, JADC2 goals risk being delayed. The consequence is that warfighters will suffer because of preconceived, institutional bias inhibiting the emergence and maturity of working capabilities and improved security. Innovation and progress become stagnated when solutions cannot escape organizational bounds, require unnecessary rework, or are impeded under the status quo by legacy practices accepted by a frozen and risk-averse middle.

CocoWow and ZARF show how the DOD can benefit from collaboration and partnerships with academic institutions such as the Naval Postgraduate School. Hundreds of students, with fleet and government experience, are waiting to solve DOD-relevant problems. The power of collaboration between uniformed forces, civilians, contractors, and academics cannot be understated.

For sailors, participation in the software life cycle means enablement, and ideally, the standup of Sailor Coders. This should be promoted, officially organized, and resourced within the Navy. A school model like the Defense Language Institute should be established to build military coding competency. Programs like Marine Coders and the Space Force's Supra Coders programs are examples on which Sailor Coders could be modeled. Sailors could participate in these cohorts and then be assigned as software sustainers.

G. FUTURE RESEARCH

This research can be extended into the operational realm for submarines and provide immediate impact. A missing component of the DevSecOps pipeline was a submarine PaaS. Using CocoWow and ZARF, pilot periods can be designed to validate the usefulness of the maintenance planning software designed by Project Blue. With the prototypes designed by the author and built with Project Blue, sailors will be given the opportunity to provide input

to replace Excel as the primary tool for maintenance planning. This will enable solutions that can help achieve the goal of getting a ship to sea on schedule.

While this research was unclassified, transferring the capabilities to higher classification networks should be explored. Further study is recommended into the requirements for submarine data using cloud environments. To develop a common operating picture between the TRFs, SRS, shipyards, and the submarines, cloud classification guidance will be required. Being able to create useful data, include it in future planning evolutions, and avoid data silos through cloud access can cause immediate benefits to submarine maintenance.

Digital modernization for the submarine environment is a priority. Understanding the current technology limitations on the submarines and the TRFs will be critical for adopting new technologies. Creating a network and infrastructure map to identify where improvements can be made would provide information on how to better invest resources to meet modernization goals. Infrastructure limitations or unknowns remains one of the greatest barriers to adoption of new technologies.

How to best manage the data produced during a maintenance period should be explored. The Bridger Project was prototyped to digitize the maintenance execution process and create useful data for analytics. Insights and efficiencies can be gained by the incorporation of machine learning and artificial intelligence into the maintenance planning process. The development of the IWS is a weeks-long process using numerous Excel spreadsheets and includes many maintenance deconfliction meetings. Thousands of man-hours are spent on the creation and updating of this document over the course of a submarine maintenance cycle. The inclusion of automation and natural language processing into current maintenance deconfliction practices can continue the push for creating a holistic and integrated solution for planning, which saves sailors time and resources.

APPENDIX A. THE BEAST CORE DIGITAL TWIN AND TECHNICAL DOCUMENT VIEWER

The following figures show the features of three-dimensional modeling for the sustainment practices of submarines with the Beast Core container. Data for this study was provided by Beast Code as part of the feasibility study for deploying cloud-based containerized applications to submarines, as well as to understand deploying digital twins under submarine resource constraints (Beast Code, n.d.).

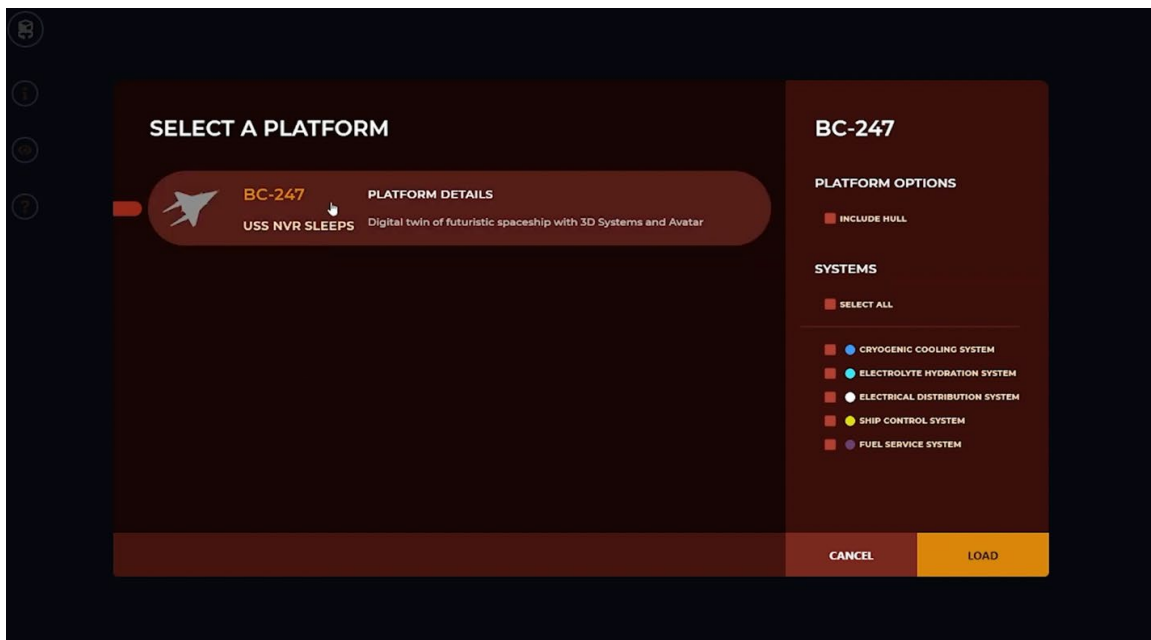


Figure 13. Selection screen

The Beast Core viewer dashboard lets the user select a platform to view. Further options include the hull silhouette and applicable shipboard systems.

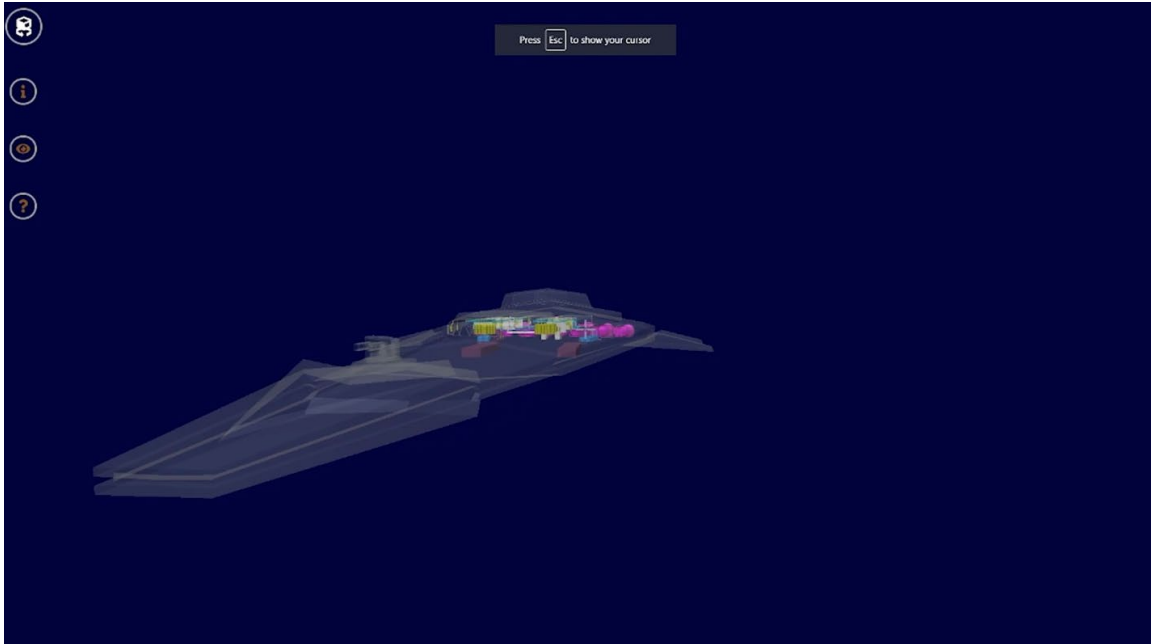


Figure 14. Full hull view of USS NEVERSLEEPS

The Beast Core viewer provides external and internal vantage points.

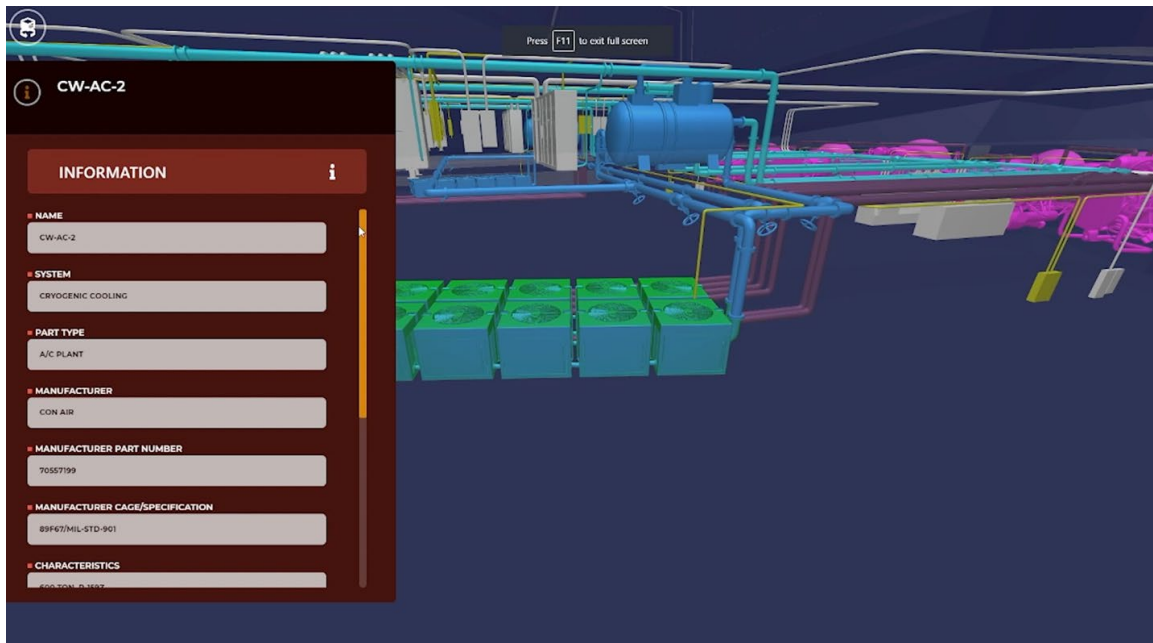


Figure 15. Component selection and specification view

Individual components can be selected and the technical data such as name, system, and characteristics can be reviewed.

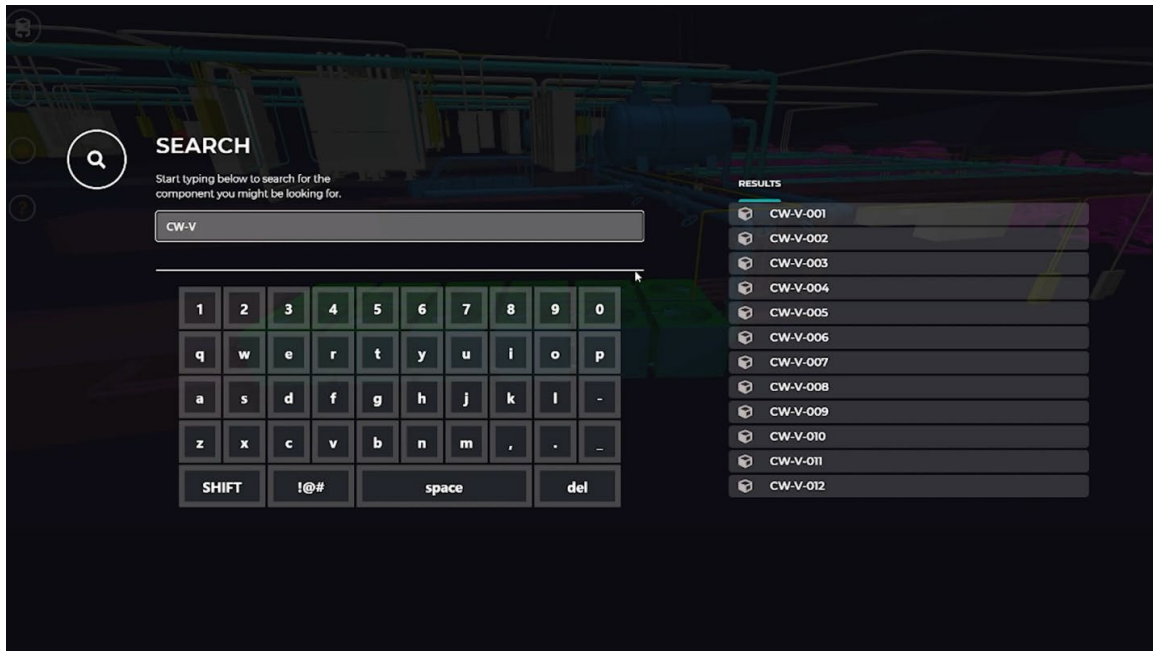


Figure 16. Search function

The search function can quickly identify components and locations.

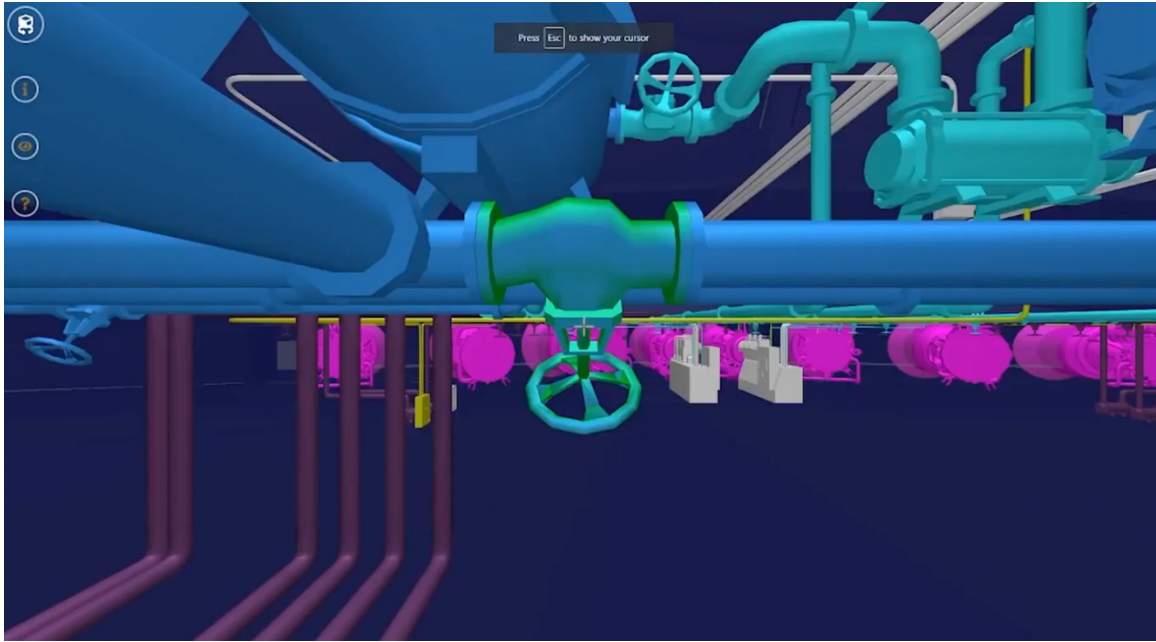


Figure 17. Search component screen zoom

Search can zoom on locations.

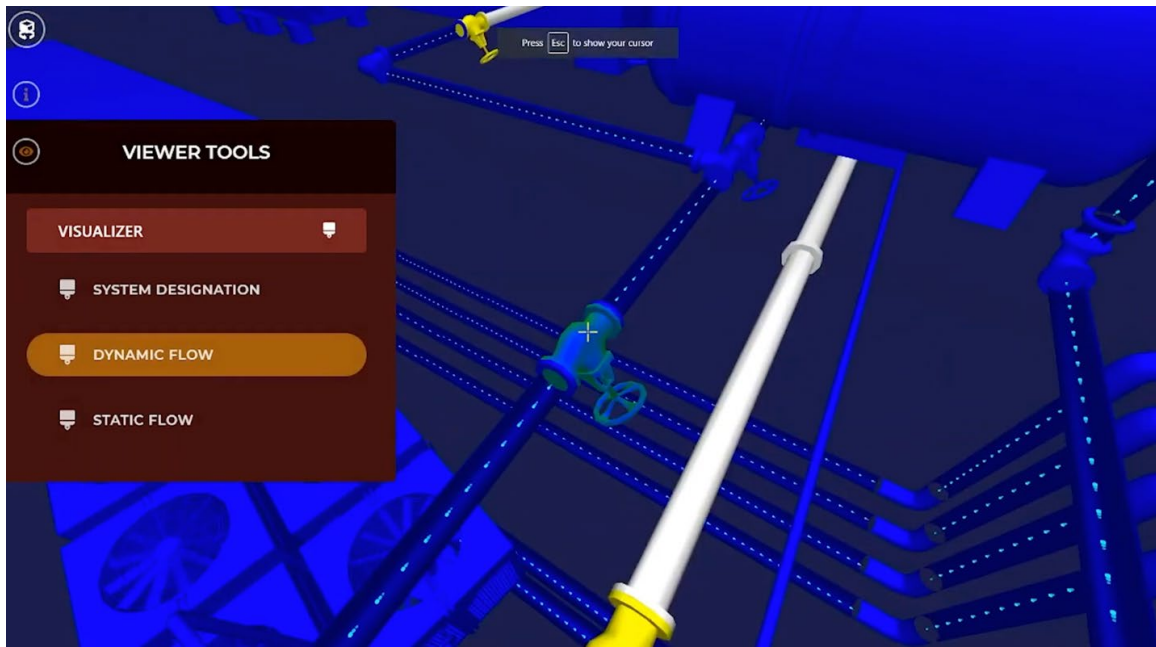


Figure 18. Flow mode visualizer

Using the System Visualizer, flow within a system can be visualized with dotted lines.

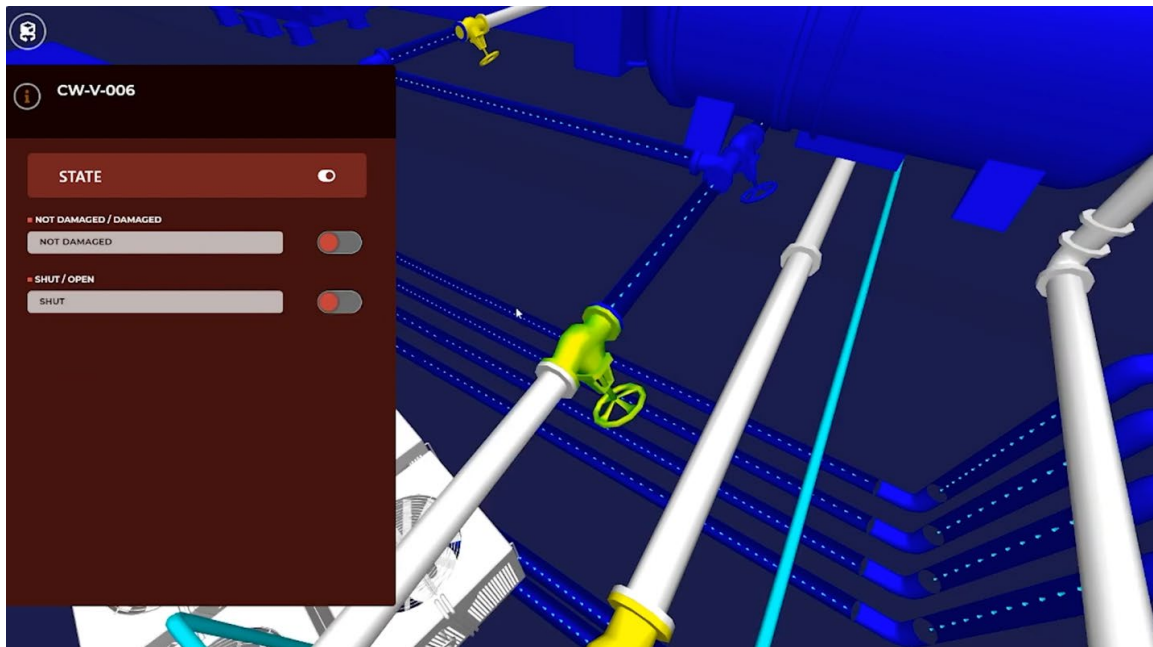


Figure 19. Component state

Individual components can be manipulated as when a valve's state is changed from open to shut.

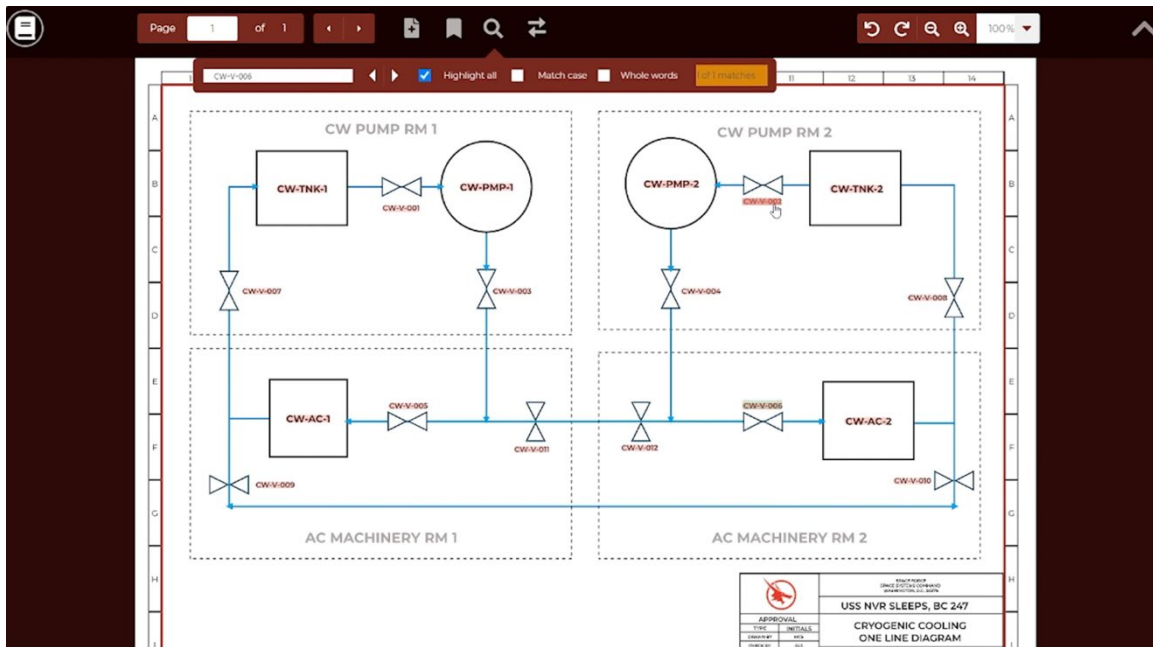


Figure 20. Two-dimensional system diagram viewer

In the technical documentation viewer, red hyperlinks indicate association within the three-dimensional model. Clicking on the hyperlink will cause the system to traverse to the component the user clicked in either the three-dimensional viewer or in Avatar mode.

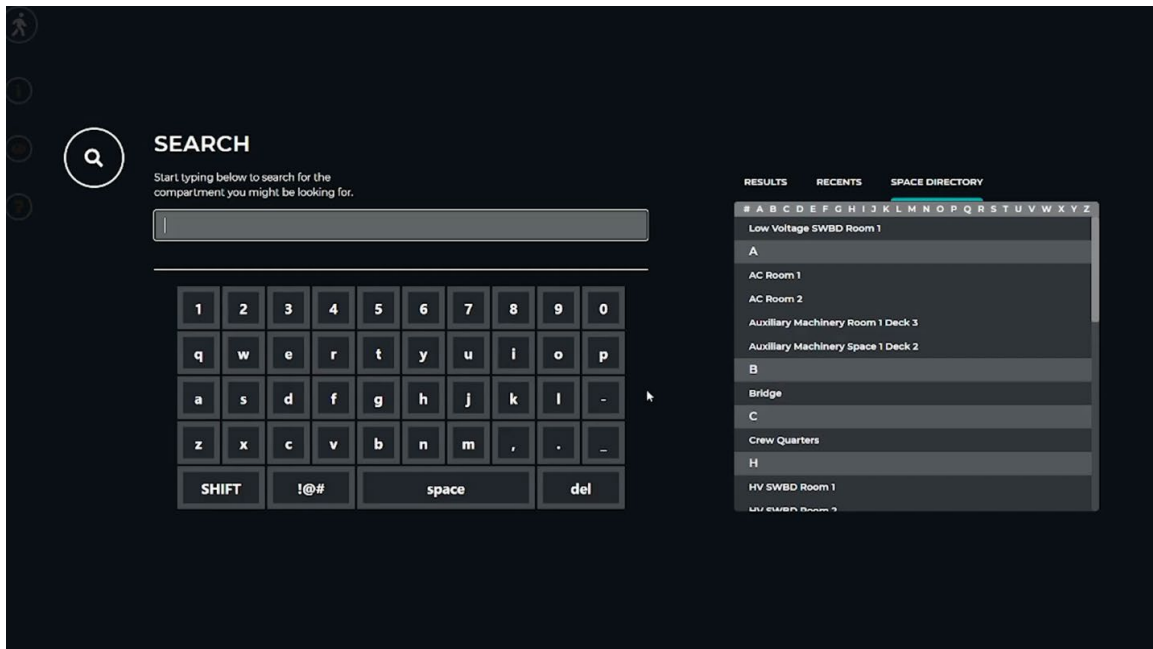


Figure 21. Space directory search

Avatar mode takes the user to the specific location.

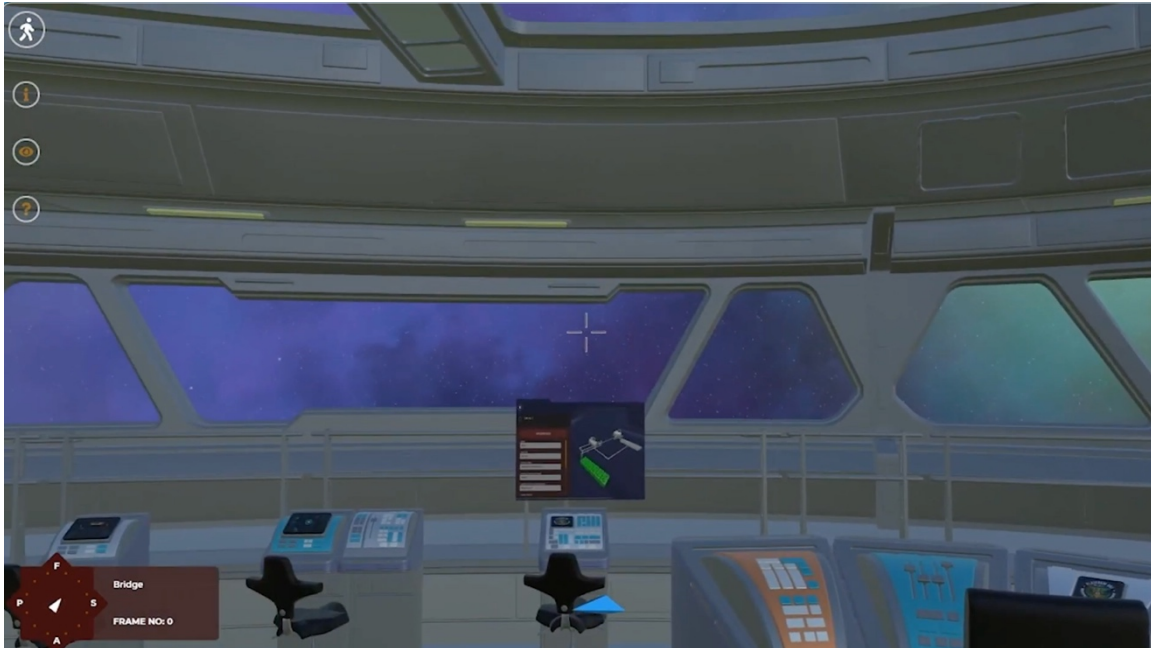


Figure 22. Avatar mode

In Avatar Mode, a user can operate as a first-person viewer and move around.

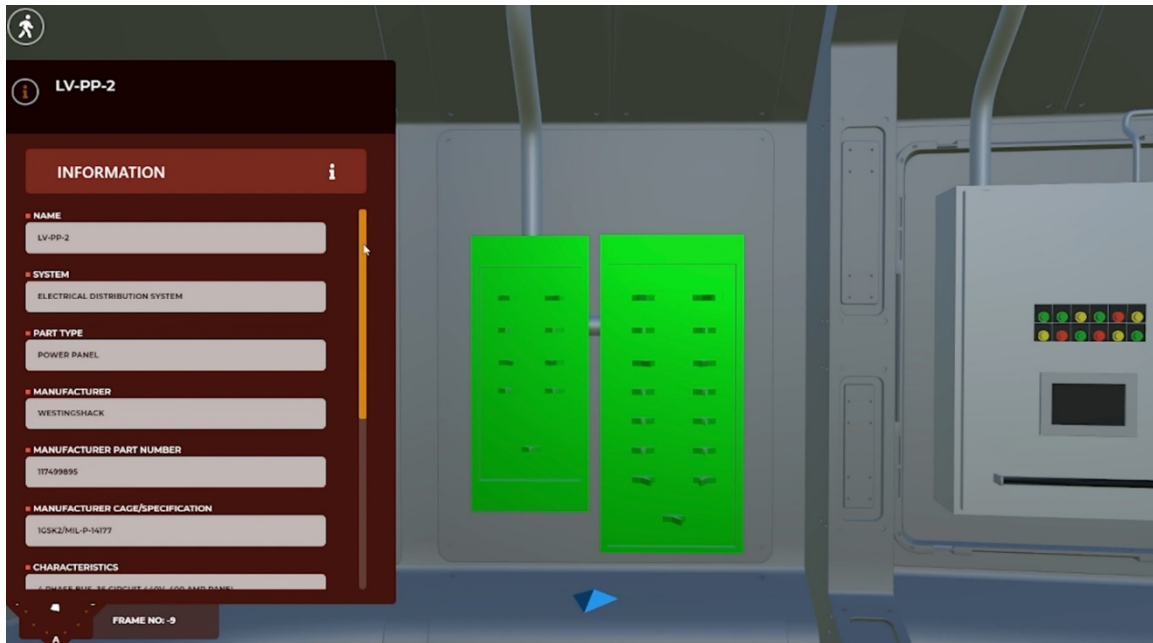


Figure 23. Avatar mode component selection

Components in Avatar Mode can be highlighted and technical information can be quickly accessed.

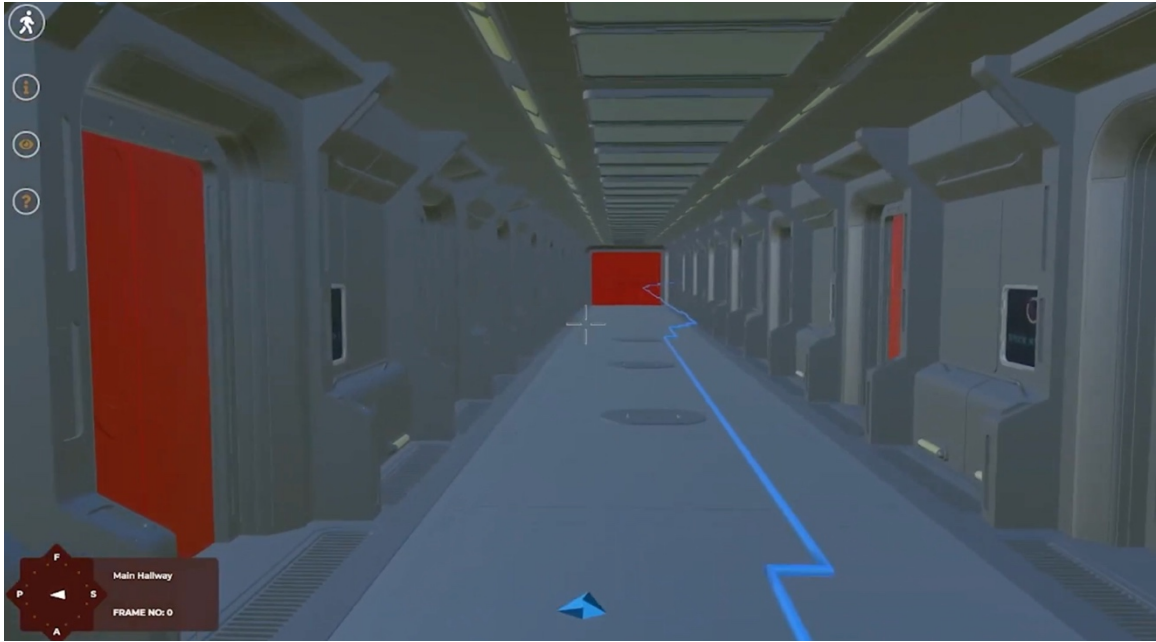


Figure 24. Avatar path display

In Avatar Mode, the most direct path to a component is provided to the user.

APPENDIX B. MASTER JOB FILE

Master Job File includes three-dimensional modeling data for maintenance prototyped by Beast Code with inputs from the author. The Master Job File is linked to the processes from the Joint Fleet Maintenance Manual and supports future design requirements for submarine platforms. This tool is for the planners who create the procedures for a submarine maintenance cycle.

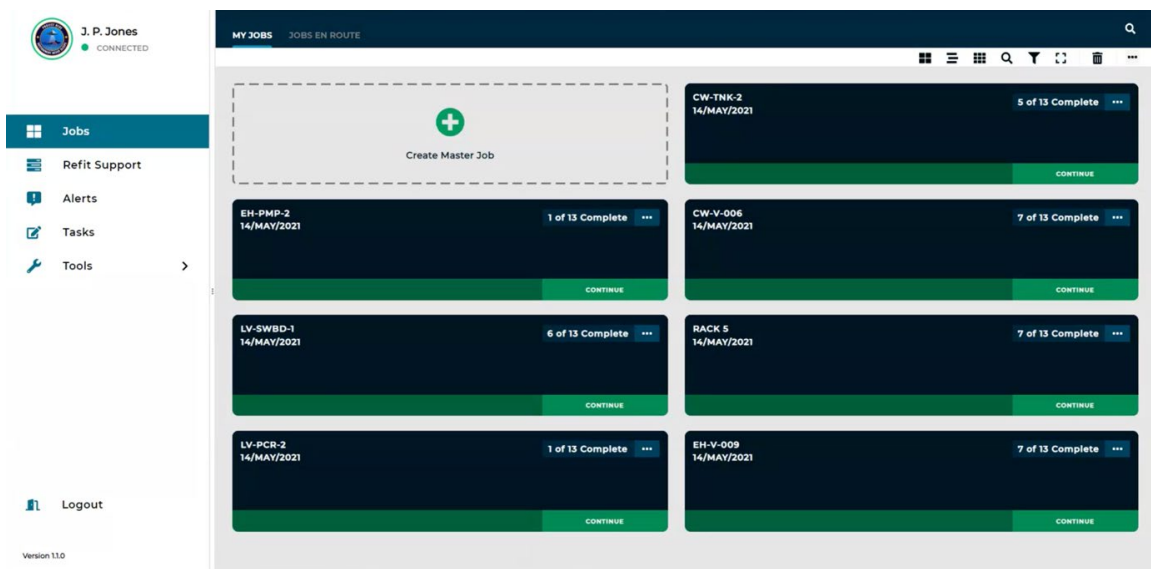


Figure 25. Master Job File dashboard

This dashboard tracks jobs created. Features include a collapsible left-hand menu, progress and review status, search functions, and a creation button to begin a new Master Job.

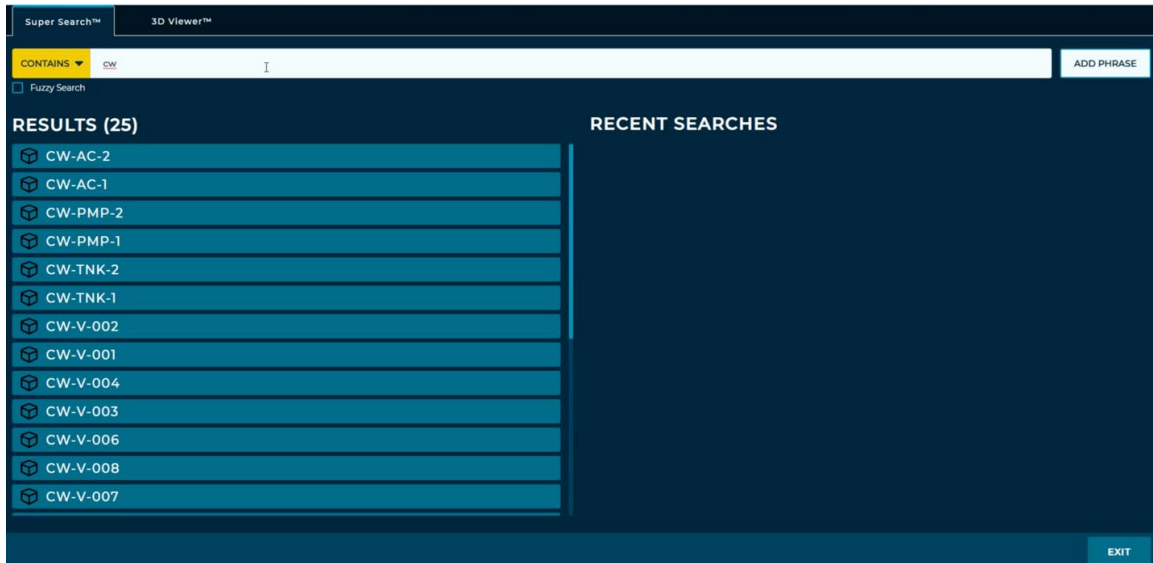


Figure 26. Component and job search

Super Search searches for components, documentation, and Master Job files. Fuzzy search provides a broader search.

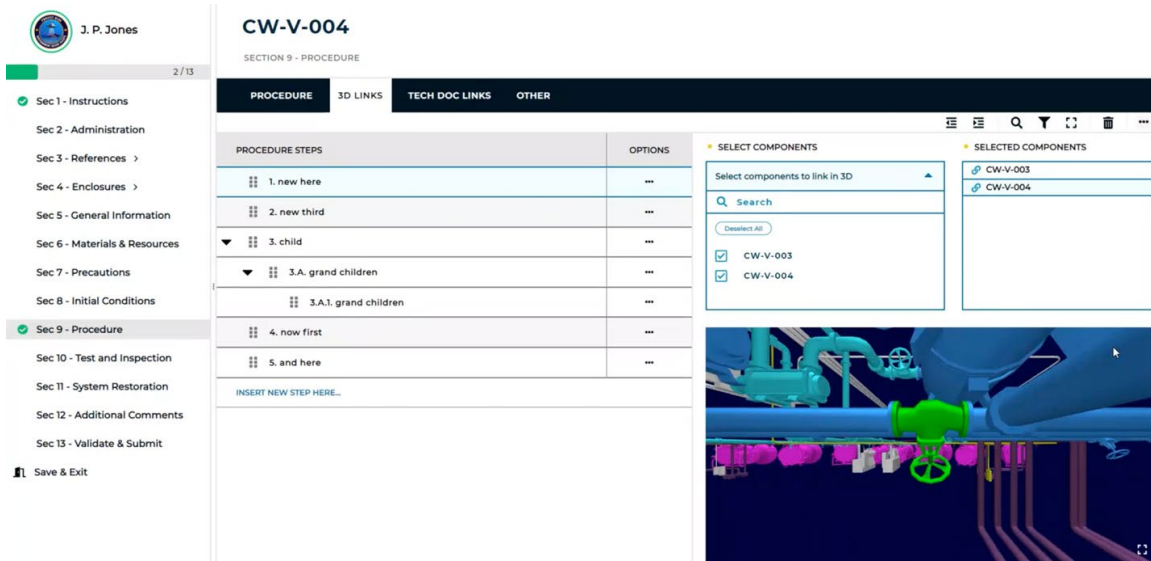


Figure 27. Procedure creation

A Master Job File follows the specifications in the Joint Fleet Maintenance Manual (JFMM). 13 sections cover the requirements for the creating work packages used in submarine maintenance; highlighted here is the Procedure section. Parent steps and child steps can be created, modified, and moved, to detail maintenance execution. The three-dimensional modeling of the Beast Core digital-twin and technical-data viewer can be linked to each procedural step.

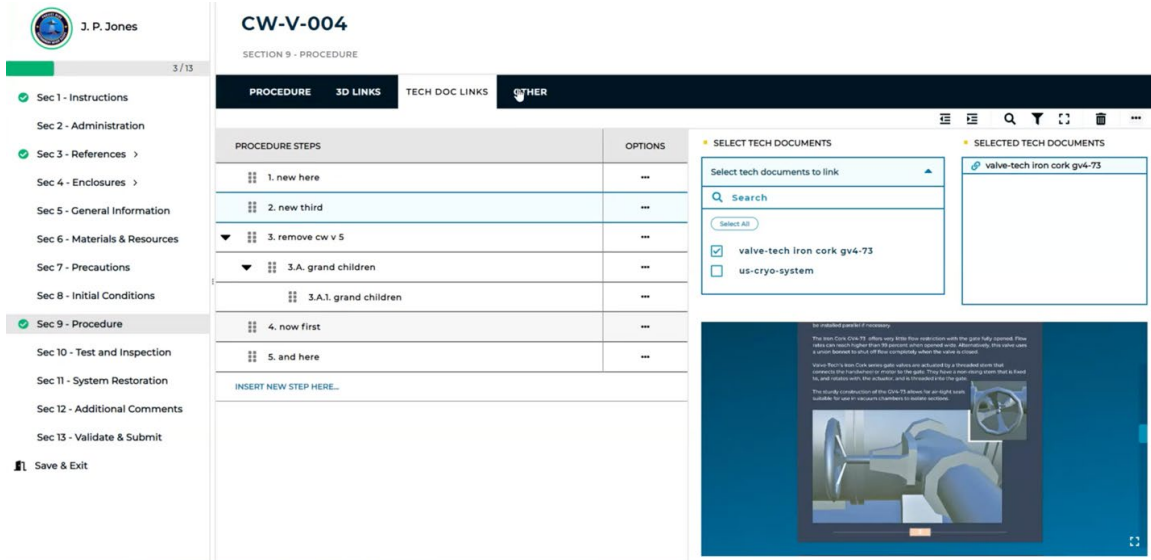


Figure 28. Master Job File reference viewer

In document-viewer mode, the references for a procedural step can be linked and displayed.

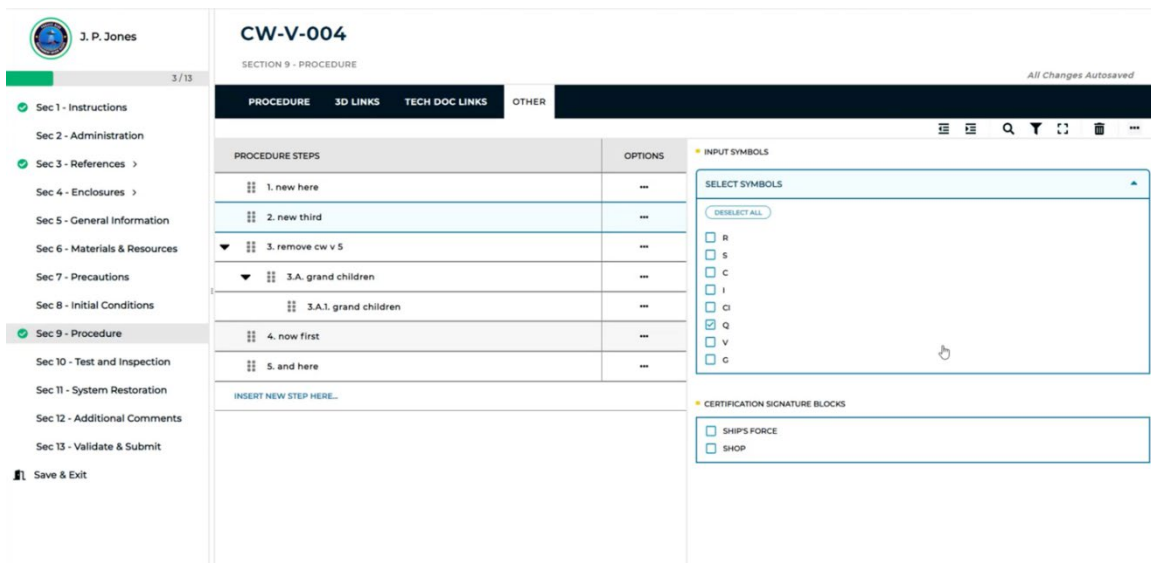


Figure 29. Critical steps and inspection points

Following guidance from the JFMM, procedural steps may require additional certification signatures at inspection.

APPENDIX C. THE BRIDGER PROJECT

The Bridger Project provides an interface to Beast Core for a submarine crew and enables planning and executing workflows during maintenance. It was prototyped by Beast Code. The prototype used agile methods for software development of the cloud-based containerized software for submarine sustainment. The capabilities and requirements were defined by the author. The graphics represent the author's experience onboard a ship.

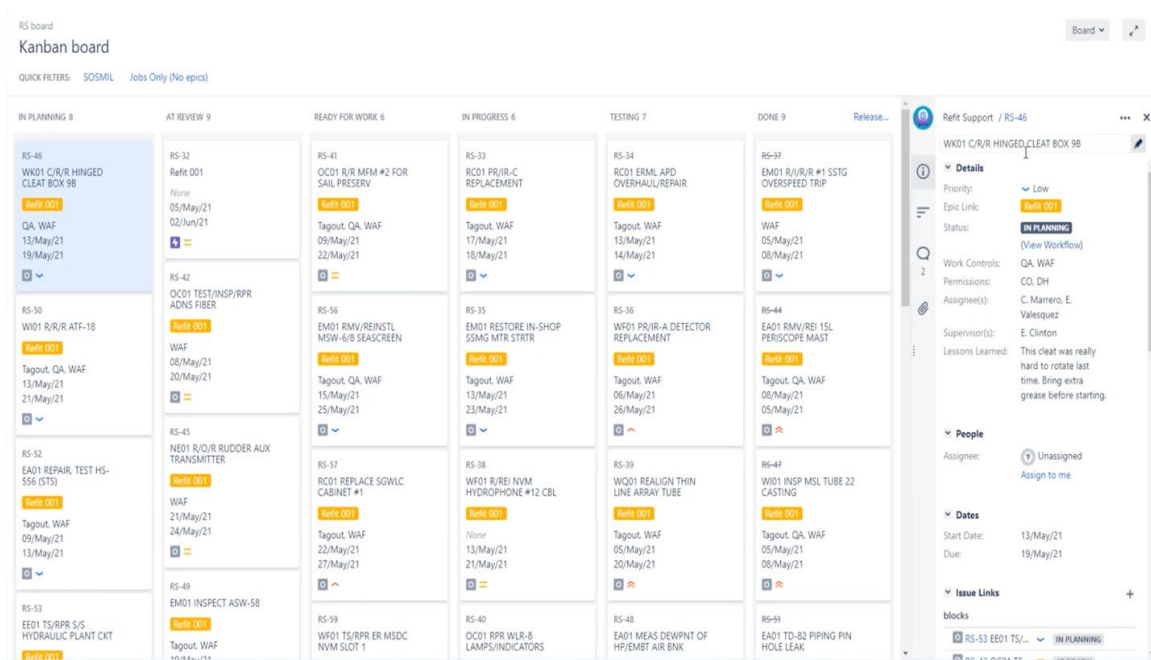


Figure 30. Job status tracking

A detailed workflow is created both automatically with LDS and manually by the team or work center responsible for completing the task. Jobs are categorized based on status of the work: in planning, at review, ready for work, in process, testing, done. As the status of jobs change and are updated, the jobs flow from left to right

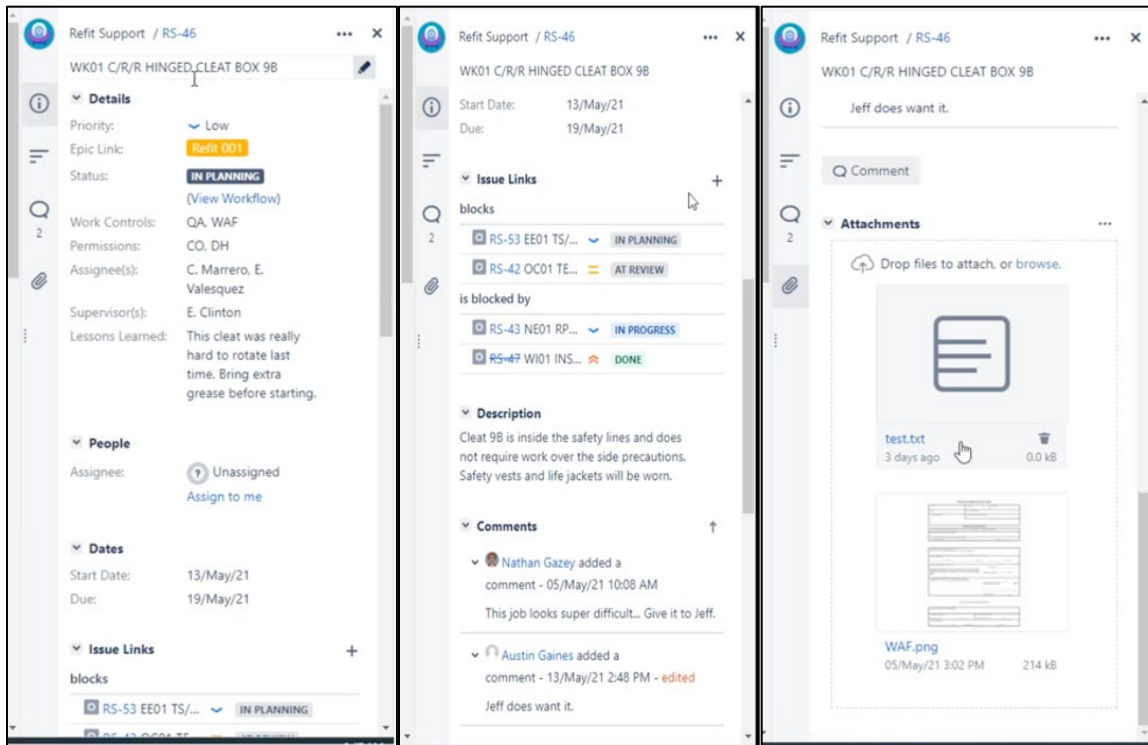


Figure 31. Individual job data

A job can be highlighted and reviewed. Attributes include assigned priority, quality assurance (QA) work, tagout required, associated Work Authorization Form (WAF), other references attached, permissions required, workers assigned, certifying supervisor, lessons learned available for review, start and expected finish dates, and any conflicting jobs that could require cross-divisional coordination. Comments can also be added at key decision points or for supervisory guidance.

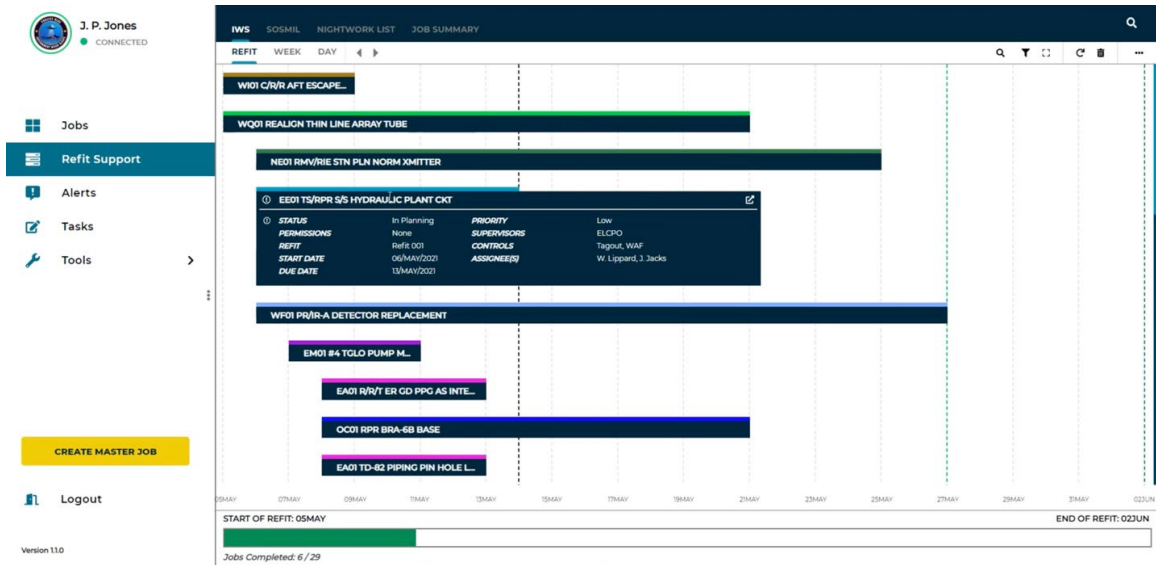


Figure 32. Integrated work schedule

The refit support module shows an Integrated Work Schedule Gantt chart viewer. All jobs are taken from the previously shown workflow board. Modes include complete refit view, week view, and day view. The divisional colors and work center are displayed for quick identification of ownership. The vertical dashed line tracks the current day for the maintenance period. A progress bar at the bottom is included for trend analysis. Validation icons indicated by an “!” note a status discrepancy to the user or a job that needs further attention. A dropdown feature for each job provides a snapshot for quick review.

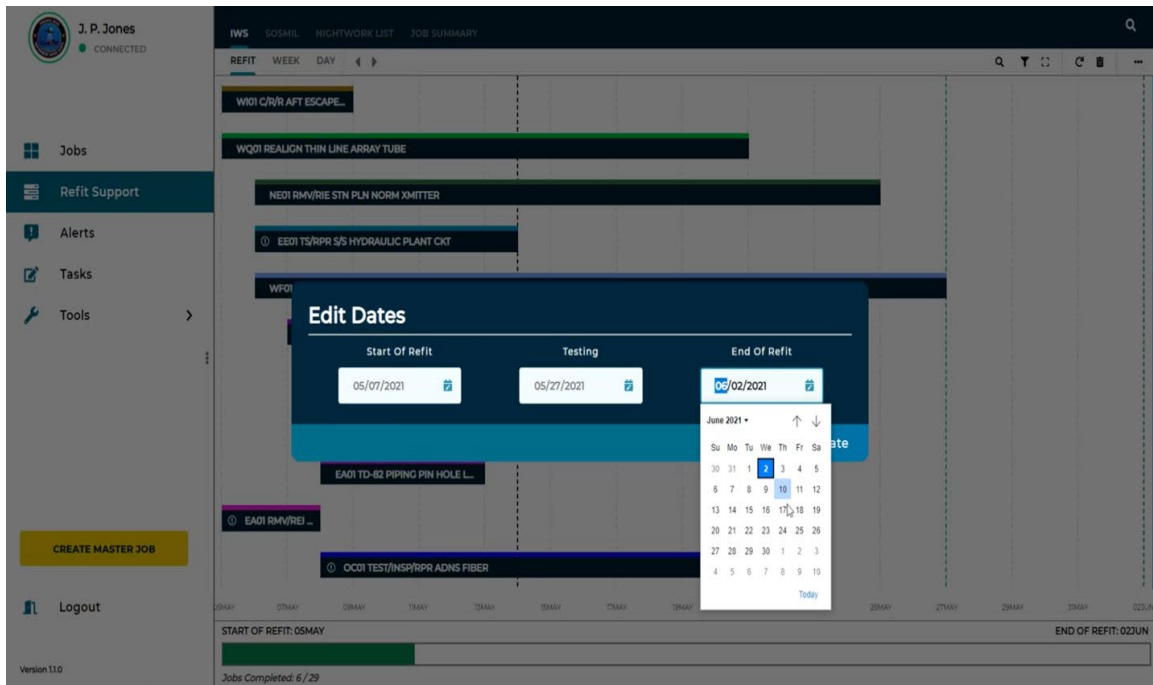


Figure 33. Timeline changes

Dates and major milestones can be adjusted as conditions within an availability change. Permissions are determined from user roles such as Commanding Officer (CO). Any changes will automatically update the Integrated Work Schedule chart.

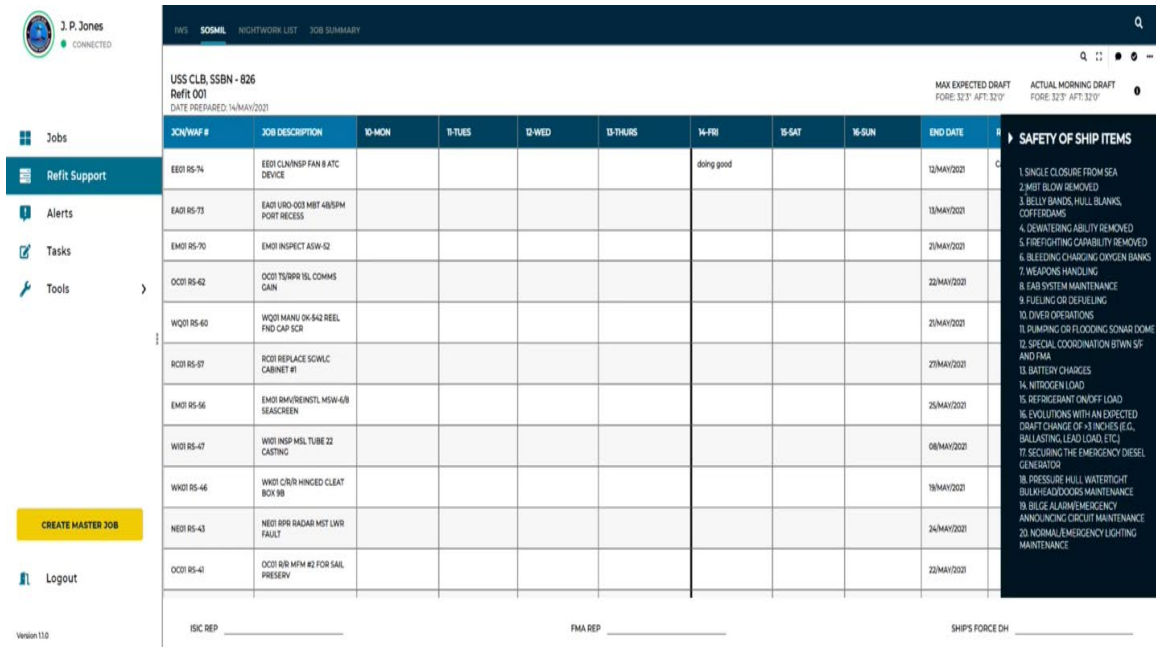


Figure 34. SOSMIL

The SOSMIL tab tracks jobs that require higher visibility and coordination between the ship and the Refit facilities. The black line indicates the current working day. The requirements are taken from the Joint Fleet Maintenance Manual and displayed or hidden based on user preference. Signatures and concurrences are tracked between commands.



J. P. Jones

CONNECTED

Jobs

Refit Support

Alerts

Tasks

Tools

Log out

FWs

SOCIAL

NIGHTWORK LIST

JOB SUMMARY

Q

Y

⇄

⌂

⌕

⌕

⌕

⌕

JOB	ASSIGNEE(S)	APPROVER(S)	TAG REQ.	WAF REQ.	STATUS	DH APPROVAL	CO APPROVAL	NOTES
EAOI URO-003 M8T 4B/5PM PORT RECESS	D. Christianson, W. Peason, K. O'Neill	Chief Criggs	N	Y	At Review	Y	Y	
EAOI R/UT ER CD PPC/AS INTERF.	A. Schiboula, W. Kolz		N	Y	Done	Y	Y	
EAOI INSP/DR #1 DCT VBR DAMPING	G. Inglefield, F. Rio	ALCPO	N	Y	At Review	N	N	
WFOI C/R/R AFT ESCAPE TRUNK	D. Stewart, N. Carol		N	N	Done	N	N	
NEOI GPS ANTENNA CHECK	E. Richard, F. Gentle	None.	N	N	Testing	Y	N	
NEOI RMV/NE STN PLN NORM XMITTER		F. Kyle	N	Y	Testing	Y	Y	
OCOI SP0 BBA 6B BASE	E. Tyler, D. Schiboula	Radio Div Chief	N	Y	Testing	Y	Y	
OCOI TS/RP/ ISL COMMS GAIN	S. Marsh, A. Edmond	Chief Kardashian	N	Y	Ready for Work	Y	N	
OCOI SP0 B3 HEAD WINDOW HEAT	W. Grossglass, F. Chaffield	Chief Miller	N	Y	Ready for Work	Y	N	
WFOI TS/RP/ ER MISC NWM SLOT1	W. Swift, A. Bahlin	Chief Buening	N	N	Ready for Work	N	N	
EMOI #4 TOLQ PUMP MECHANICAL SEAL	S. Dobson, J. Ledestich	MLCPO	N	Y	Done	Y	N	
EMOI SSW 20 R/UT	S. Fox, E. Limoges	MLCPO	N	Y	In Planning	N	N	
EMOI OVAL MS-19 GLOBE VALVE	G. McGregor, Y. Timothy	RLCPO	N	Y	In Planning	N	N	
WFOI C/R/R HINGED CLEAT BOX 9B	C. Marnes, E. Velequez	E. Clinton	N	Y	In Planning	Y	Y	
RCOI BSM, APD OVERHAUL/REPAIR	G. Haley, S. Jay	E. Elliot	N	Y	Testing	Y	Y	

Version 110

The Nightwork list is the source for Ship's Duty Officers (SDOs) and Production Officers to authorize maintenance. Jobs are entered that require permission of the Department Head and Captain before starting work. A notes feature allows additional comments, such as "Call before starting."

5/14/2021

Royal

1 page

	JOB	ASSIGNEE(S)	APPROVER(S)	TAG REQ.	WAF REQ.	STATUS	DH APPROVAL	CO APPROVAL	NOTES
	EA01 URO-003 MBT 4B/5PM PORT RECESS	D. Christianson, W. Pearson, K. O'Neill	Chief Griggs	N	Y	At Review	Y	Y	
EA01	EA01 R/R/T ER GD PPG AS INTERF.	A. Schiboula, W. Kolz		N	Y	Done	Y	Y	
EA01	EA01 INSP/RPR #1 DCT VIBR DAMPING	G. Ingleside, F. Rio	ALCPO	N	Y	At Review	N	N	
EA01	WI01 C/R/R AFT ESCAPE TRUNK	D. Stewart, N. Carrol		N	N	Done	N	N	
EA01	NE01 GPS ANTENNA CHECK	E. Richard, F. Gentile	None.	N	N	Testing	Y	N	
	NE01 RMV/RIE STN PLN NORM XMITTER		F. Kyle	N	Y	Testing	Y	Y	
WI01	OC01 RPR BRA-6B BASE	E. Tyler, D. Schiboula	Radio Div Chief	N	Y	Testing	Y	Y	
NE01	OC01 TS/RPR ISL COMMS GAIN	S. Marsh, A. Edmond	Chief Kardashian	N	Y	Ready for Work	Y	N	
NE01	OC01 RPR 83 HEAD WINDOW HEAT	W. Grossglass, F. Chrifield	Chief Miller	N	Y	Ready for Work	Y	N	
OC01	WF01 TS/RPR ER MSDC NVM SLOT 1	W. Swift, A. Rahim	Chief Buening	N	N	Ready for Work	N	N	
OC01	EM01 #4 TCLO PUMP MECHANICAL SEAL	S. Dobson, J. Ledestich	MLCPO	N	Y	Done	Y	N	
OC01	EM01 SSW-20 R/R/T	S. Fox, E. Limoges	MLCPO	N	Y	In Planning	N	N	
OC01	EM01 OVHL MS-19 CLOBE VALVE	G. McGregor, Y. Timothy	RLCPO	N	Y	In Planning	N	N	
WF01	WK01 C/R/R HINGED CLEAT BOX 9B	C. Marrero, E. Valesquez	E. Clinton	N	Y	In Planning	Y	Y	
EM01	RC01 ERM L APD OVERHAUL/REPAIR	G. Holey, S. Jay	E. Elliot	N	Y	Testing	Y	Y	

Print

Destination

Save as PDF

Pages

All

Layout

Landscape

More settings

localhost:1337?ref=support@network-ist

1/1

Save

Cancel

Figure 36. Print nightwork

Once approved this Nightwork document is printed and routed for signature at the daily supervisor meeting. Once authorized it is routed to both the Ship's Duty Officer and the Engineering Duty Officer.

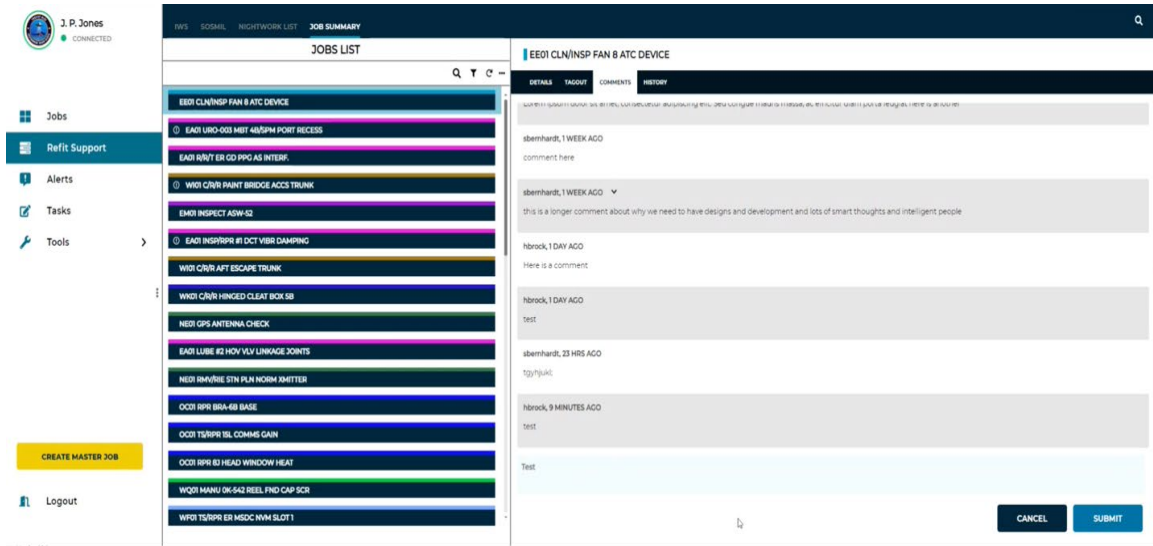


Figure 37. Job summary

The Job-Summary tab logs comments and lessons-learned associated with the work and attests to completion of the job. Upon completion or delay, maintainers input comments about the status of a job and any related information. This information is used in the Job Summary Report to create the End of Refit Summary Report.

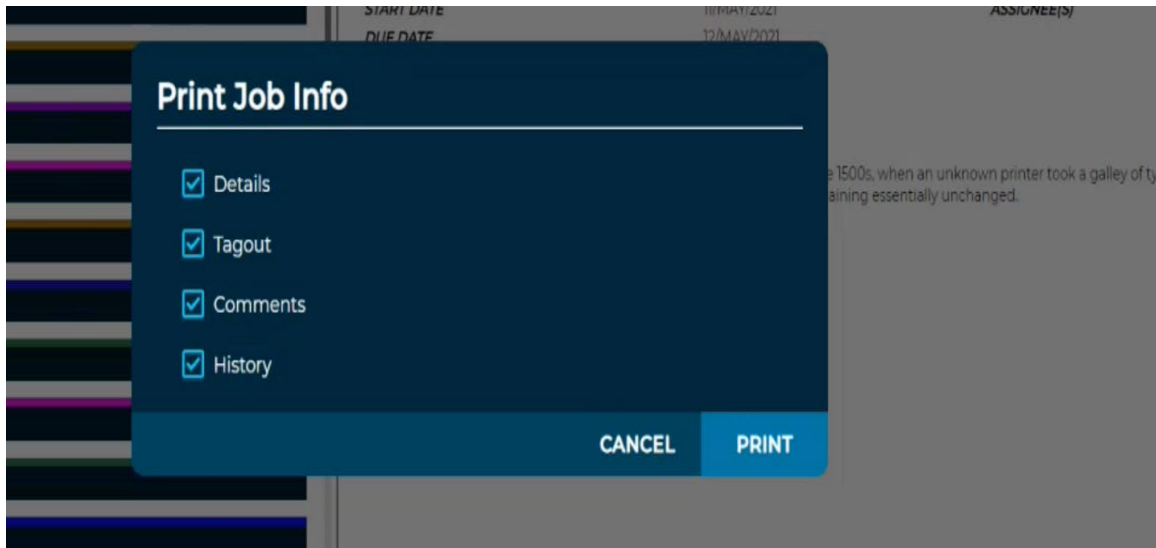


Figure 38. Refit reports

Options to include in the End of Refit Report include details, tagouts, comments from the job site, and history of changes.

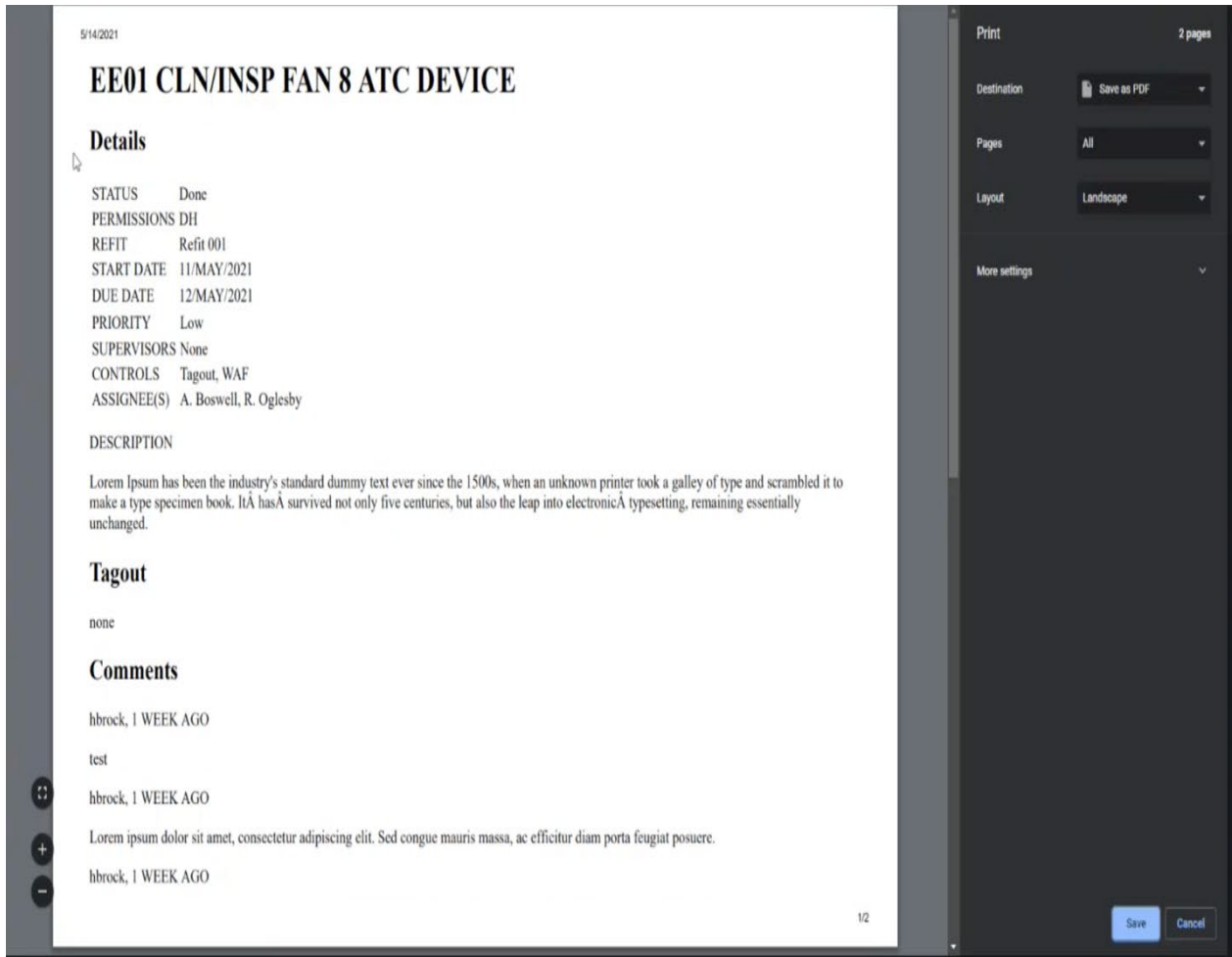


Figure 39. Print reports

A detailed report is generated from information logged for each job executed during a maintenance and availability period. This is forwarded for review by the TRFs and saved for documentation of Lessons Learned.

APPENDIX D. ZARF

The following are the procedural steps and outputs for deploying ZARF to an operating system. ZARF was created in collaboration with Defense Unicorns and Project Blue (Github, n.d.) The author provided architecture and design inputs for the submarine use case and testing. This is representative of the deployment tests done at the Naval Undersea Warfare Center and its sandbox testing. Key takeaways are italicized.

[illegible]

Figure 40. Initial ZARF interface

A. PHASE 1: PLATFORM-AS-A-SERVICE INSTALL

ZARF initialization of the platform-as-a-service occurs with the following command.

```
~?./zarf init
```

ZARF asks the user if they would like to generate their own TLS certificates.

```
~? Will Zarf be generating a TLS chain or importing an existing ingress cert?  
Generate TLS chain with an ephemeral CA
```

ZARF asks for the user network configuration for their environment.

```
~? Enter a host DNS entry or IP Address for the cluster ingress localhost
```

ZARF makes initial checks to verify its system initializations are correct.

```
INFO[0007] Preflight check: validating os type  
INFO[0007] Preflight check: validating AMD64 arch  
INFO[0007] Preflight check: validating user is root  
INFO[0007] Preflight check: validating hostname  
INFO[0007] Installing K3s  
INFO[0007] Creating temp path  
path=/tmp/zarf-050516502  
INFO[0007] Extracting the package, this may take a few moments
```

ZARF displays what is installing on the system and the versions.

```
kind: ZarfInitConfig  
metadata:  
  name: ""  
  description: ""  
  version: ""  
uncompressed: false  
package:  
  terminal: runner-esz8b4jn-project-6178-concurrent-0dbhqj  
  user: root  
  timestamp: Wed, 06 Oct 2021 19:49:15 +0000  
  data: []  
components:  
  - name: k3s  
  description: Install K3s  
  default: false  
  required: true  
manifests: assets/manifests/common
```

```

images:
- docker.io/rancher/coredns-coredns:1.8.3
- docker.io/rancher/klipper-helm:v0.5.0-build20210505
- docker.io/rancher/klipper-lb:v0.2.0
- docker.io/rancher/library-busybox:1.32.1
- docker.io/rancher/library-traefik:2.4.8
- docker.io/rancher/local-path-provisioner:v0.0.19
- docker.io/rancher/metrics-server:v0.3.6
- docker.io/rancher/pause:3.1
repos: []
charts: []
files:
- source: https://github.com/k3s-io/k3s/releases/download/v1.21.2+k3s1/k3s
  shasum: 5097d515e220f8e97ab13c56cb9142ee4526b4c9eade5ed098e2906c1db2a163
  target: /usr/local/bin/k3s
  executable: true
- source: assets/scripts/k3s-remove.sh
  shasum: ""
  target: /usr/local/bin/k3s-remove.sh
  executable: true
- source: assets/scripts/k3s.service
  shasum: ""
  target: /etc/systemd/system/k3s.service
  executable: false
- source: assets/misc/registries.yaml
  shasum: ""
  target: /etc/rancher/k3s/registries.yaml
  executable: false
- name: container-registry
description: ""
default: false
required: true
manifests: assets/manifests/registry
images:
- registry1.dso.mil/ironbank/opensource/docker/registry-v2:2.7.1
repos: []
charts:
- name: docker-registry
  url: https://helm.twun.io
  version: 1.10.1
files: []
- name: management
description: Add the K9s terminal-based K8s UI for cluster management
default: true
required: false
manifests: ""
images: []
repos: []
charts: []
files:
- source: https://zarf-public.s3-us-gov-west-1.amazonaws.com/k9s_Linux_x86_64_v0_24_11
  shasum: 18a5a33bbf58cb228e56a03380dcb6b9bb8624acab4ff63deb7364dc15d3c03f
  target: /usr/local/bin/k9s
  executable: true
- source: assets/misc/k9s-theme.yaml
  shasum: ""
  target: /root/.k9s/skin.yml
  executable: false

```

```

- name: logging
description: Add Promtail, Grafana and Loki (PGL) to this cluster for log monitoring.
default: true
required: false
manifests: assets/manifests/logging
images:
- grafana/loki:2.2.0
- grafana/promtail:2.1.0
- grafana/grafana:7.5.0
- kiwigrid/k8s-sidecar:0.1.209
repos: []
charts:
- name: loki-stack
  url: https://grafana.github.io/helm-charts
  version: 2.4.1
files: []
- name: gitops-service
description: Add Gitea for serving gitops-based clusters in an airgap
default: false
required: false
manifests: assets/manifests/gitops
images:
- gitea/gitea:1.13.7
repos: []
charts:
- name: gitea
  url: https://dl.gitea.io/charts
  version: 2.2.5
files: []

```

ZARF asks the operator to confirm the package.

```

~? Deploy this Zarf package? Yes

INFO[0202] Loading dynamic config
path=/tmp/zarf-050516502/zarf.yaml
INFO[0202] Deploying Zarf component
name=k3s
INFO[0202] Loading files for local install
INFO[0203] Loading images for local install
INFO[0203] Copying file Destination=/var/lib/rancher/k3s/agent/images/images-k3s.tar
Source=/tmp/zarf-050516502/components/k3s/images-component-k3s.tar
INFO[0203] Loading manifests for local install, this may take a minute or so to reflect in k3s
INFO[0203] Processing manifest file
path=/tmp/zarf-050516502/components/k3s/manifests/traefik-tls.yaml
INFO[0203] Copying
file
Destination=/var/lib/rancher/k3s/server/manifests
Source=/tmp/zarf-050516502/components/k3s/manifests
INFO[0203] Deploying Zarf component
name=container-registry
INFO[0203] Loading charts for local install
INFO[0203] Copying file Destination=/var/lib/rancher/k3s/server/static/charts/docker-registry-
1.10.1.tgz Source=/tmp/zarf-050516502/components/container-registry/charts/docker-registry-1.10.1.tgz
INFO[0203] Loading images for local install

```

```
INFO[0203] Copying file           Destination=/var/lib/rancher/k3s/agent/images/images-container-registry.tar
Source=/tmp/zarf-050516502/components/container-registry/images-component-container-registry.tar
INFO[0203] Loading manifests for local install, this may take a minute or so to reflect in k3s
INFO[0203] Processing manifest file
path=/tmp/zarf-050516502/components/container-registry/manifests/registry.yaml
INFO[0203] Copying file           Destination=/var/lib/rancher/k3s/server/manifests
Source=/tmp/zarf-050516502/components/container-registry/manifests
```

ZARF asks the operator if they want the Kubernetes management tool K9s.

```
~? Deploy the management component? Yes
INFO[0240] Deploying Zarf component
name=management
INFO[0240] Loading files for local install
```

ZARF asks the operator if they want the Promtail, Loki and Grafana logging services.

```
? Deploy the logging component? Yes
INFO[0246] Deploying Zarf component
name=logging
INFO[0246] Loading charts for local install
INFO[0246] Copying file           Destination=/var/lib/rancher/k3s/server/static/charts/loki-stack-2.4.1.tgz
Source=/tmp/zarf-050516502/components/logging/charts/loki-stack-2.4.1.tgz
INFO[0246] Loading images for local install
INFO[0246] Copying file           Destination=/var/lib/rancher/k3s/agent/images/images-logging.tar
Source=/tmp/zarf-050516502/components/logging/images-component-logging.tar
INFO[0246] Loading manifests for local install, this may take a minute or so to reflect in k3s
INFO[0246] Processing manifest file
path=/tmp/zarf-050516502/components/logging/manifests/pgl-stack.yaml
INFO[0246] Copying file           Destination=/var/lib/rancher/k3s/server/manifests
Source=/tmp/zarf-050516502/components/logging/manifests
```

ZARF asks the operator if they want the Gitops-service component.

```
? Deploy the gitops-service component? No
INFO[0260] Cleaning up temp files
systemctl [daemon-reload]
systemctl [enable --now k3s]
INFO[0265] Creating kube config symlink
INFO[0265] Loading secret
Cert=zarf-pki/zarf-server.crt
Name=tls-pem
Namespace=kube-system
INFO[0265] Adding Ephemeral CA to the host root trust store
INFO[0265] Copying file
Destination=/usr/local/share/ca-certificates/extra/zarf-ca.crt
Source=zarf-pki/zarf-ca.crt
```

ZARF has asked the operator about the services they want and starts generating the TLS certificates for network connections.

update-ca-certificates

```
Updating certificates in /etc/ssl/certs...
0 added, 0 removed; done.
Running hooks in /etc/ca-certificates/update.d...
done.
Ephemeral CA below and saved to zarf-pki/zarf-ca.crt
```

```
-----BEGIN CERTIFICATE-----
MIIDVDCCAjygAwIBAgIQWwS1FZGXfSvyUM0t2v3qJLjANBgkqhkiG9w0BAQsFADBE
MRUwEwYDVQQKEwxaYXJmIENsdXN0ZXIxKzApBgNVBAMTIllphcmYgUHJpdmdF0ZSBD
ZXJ0aWZpY2F0ZSBBdXR0b3JpdHkwHhcNMjExMTA5MDQyODE0WmcNMjExMTA5MDQy
ODE0WjBEMRUwEwYDVQQKEwxaYXJmIENsdXN0ZXIxKzApBgNVBAMTIllphcmYgUHJp
dmF0ZSBDZXJ0aWZpY2F0ZSBBdXR0b3JpdHkwggEiMA0GCSqGSIb3DQEBAQUAA4IB
DwAwggEKAoIBAQCWxJ2gVbQei2r7fgA3LKlCJIL0IBQStj4uPmzB4SJfETJKIAc
I9xrXPlxSpLdJ35ONI/CzlzBCQLXVMmDm9JNerefhTVBwq1R1RTEjMj7VvZfL8g
NwGh051s3Gx7b1j2fzhskUXpA0RLUAuhX4jS90M8jGSzfW+1UKkFg3fgsm0uPz9
npawWbU00P71xebiPDdpzizJwj4QXmxXTfdOIXiKS0tWJS3/8KOGzDJr4OJYja56
sUzNK+FZmaGf+AZ3OEuV8mRx8Blt/bh7LMsVGo6S5+JzIz7NaLeXmLb4lg74VoKZ
+dJUW8h9VQDQ/Mko5Xn3KPFHgvYuRXk1pkuzAgMBAAGjQjBAMA4GA1UdDwEB/wQE
AwIcPDApBgNVHRMBAf8EBTADAQH/MB0GA1UdDgQWBBTqAFWZ/SEOINZau9foufEd
iktqnjANBgkqhkiG9w0BAQsFAAOCAQEAdkhpftgYQYZ24GcMV2qToOAIKa67tOa
uSTC9sZ2H30tgQpNT0yOqmYgYiyTv5Mpfgi3Uc8gzHQZunsFCj1XGH1l/tJkyqas
PWTYTFK19XUM1jg58spM6KgrDfQ07BLi+SO2RntjsBjdhLo6xfdTcFq+Bnu90zWe
+Psd7ZlvOXbgziTyWxPCuNzbwMQpHrh1IAS0vlqes5EekL/3QndfDf+6BZS2vjGE
0eW696IP3xTWfGylBGeFnXIHpErGkV3Ru6YCFcVPACFV4b77YvWRaul2ShqspmtB
7JEV+49CQS9+gHjWku1jqwoYU1otAalCVpLcc69A80/zm+Bh1dycMQ==
-----END CERTIFICATE-----
```

ZARF tells the operator that k9s is available and provides the one-time use credentials for signing into the services.

```
INFO[0266] Installation complete. You can run "/usr/local/bin/k9s" to monitor the status of the deployment.
WARN[0266] Credentials stored in ~/.git-credentials
Gitea Username (if installed)=zarf-git-user
Grafana Username=zarf-admin
Password (all)=Z3WnUtoZL69i-WEiPYK2vzqoGT89
```

ZARF has successfully completed the initial PaaS installation. K9s can verify that all services are available and running.

```
~?root@zarf-test:/home/vagrant# k9s
```

The screenshot shows the K9s terminal interface. At the top, it displays context and cluster information: Context: default, Cluster: default, User: default. Below this, it shows K9s and K8s versions, CPU usage (2%), and memory usage (25%). A list of keyboard shortcuts is provided on the right, including actions like Attach, Delete, Describe, Edit, Help, Kill, Logs, Previous, Port-Forward, Shell, Show PortForward, and YAML. The main part of the screen displays a table of pods for the 'all' namespace, with 13 pods listed. The table columns include Namespace, Name, Phase, Ready, Restarts, Status, CPU, Memory, CPU Request, CPU Limit, Memory Request, Memory Limit, IP, Node, and Age. The pods are sorted by age, with the oldest pod at the top.

Namespace	Name	Phase	Ready	Restarts	Status	CPU	Memory	CPU Request	CPU Limit	Memory Request	Memory Limit	IP	Node	Age
kube-system	coredns-7448498f4d-qcnlh	●	1/1	0	Running	5	12	5	n/a	17	7	10.42.0.2	zarf-test	4m29s
kube-system	helm-install-traefik-cntnf	●	0/1	1	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.4	zarf-test	4m29s
kube-system	helm-install-traefik-crd-lps6z	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.6	zarf-test	4m29s
kube-system	local-path-provisioner-5ff76fc89d-4wc7d	●	1/1	0	Running	2	7	n/a	n/a	n/a	n/a	10.42.0.3	zarf-test	4m29s
kube-system	metrics-server-86cbb8457f-tnckh	●	1/1	0	Running	1	13	n/a	n/a	n/a	n/a	10.42.0.8	zarf-test	4m29s
kube-system	svclb-traefik-kz47q	●	2/2	0	Running	0	2	n/a	n/a	n/a	n/a	10.42.0.14	zarf-test	4m10s
kube-system	traefik-97b44b794-fkql2	●	1/1	0	Running	1	21	n/a	n/a	n/a	n/a	10.42.0.13	zarf-test	4m10s
logging	helm-install-loki-xz7br	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.5	zarf-test	4m29s
logging	loki-0	●	1/1	0	Running	2	19	n/a	n/a	n/a	n/a	10.42.0.9	zarf-test	4m11s
logging	loki-grafana-c55d7d969-9x4dj	●	1/1	0	Running	1	19	n/a	n/a	n/a	n/a	10.42.0.10	zarf-test	4m11s
logging	loki-promtail-brlt2	●	1/1	0	Running	16	29	n/a	n/a	n/a	n/a	10.42.0.11	zarf-test	4m11s
registry	helm-install-kevin-ddwzg	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.7	zarf-test	4m29s
registry	kevin-docker-registry-8c5bbb65-wnnbg	●	1/1	0	Running	1	5	1	0	1	0	10.42.0.15	zarf-test	4m11s

B. PHASE 2: APPLICATION AND DATA INSTALL

ZARF begins the deployment of the mission application.

```
~?root@zarf-test:/home/vagrant# ./zarf package deploy
```

ZARF asks what package should be deployed: A tab over features allows for autofill from the current folder of the operator.

```
? Choose or type the package file zarf-package-appliance-demo-doom.tar.zst
INFO[0005] Creating temp path
path=/tmp/zarf-247049737
INFO[0005] Extracting the package, this may take a few moments
```

ZARF displays the configuration file for the container for the operator to verify. In this example, the game DOOM will be deployed from the Iron Bank.

```
kind: ZarfPackageConfig
```

```
metadata:
name: appliance-demo-doom
description: Demo Zarf appliance mode with doom game
version: ""
uncompressed: false
package:
terminal: Bridgers-MacBook-Pro.local
user: Bridger
timestamp: Fri, 05 Nov 2021 14:15:51 -0700
data: []
components:
- name: baseline
description: ""
default: false
required: true
manifests: manifests
images:
- registry.dso.mil/platform-one/big-bang/apps/product-tools/zarf/game:doom
repos: []
charts: []
files: []
```

ZARF asks the user if this is the correct package.

```
~? Deploy this Zarf package? Yes
```

ZARF deploys the elements specified and displays successful steps.

```
INFO[0012] Loading dynamic config
path=/tmp/zarf-247049737/zarf.yaml
INFO[0012] Deploying Zarf component
name=baseline
INFO[0012] Loading images for local install
INFO[0012] Loading images for gitops service transfer
INFO[0012] Loading images
INFO[0012] Updating image
image="registry.dso.mil/platform-one/big-bang/apps/product-tools/zarf/game:doom"
INFO[0012] 127.0.0.1/platform-one/big-bang/apps/product-tools/zarf/game:doom
INFO[0012] Loading manifests for local install, this may take a minute or so to reflect in k3s
INFO[0012] Processing manifest file
path=/tmp/zarf-247049737/components/baseline/manifests/game.yaml
INFO[0012] Copying file Destination=/var/lib/rancher/k3s/server/manifests
Source=/tmp/zarf-247049737/components/baseline/manifests
INFO[0012] Cleaning up temp files
```

K9s verifies that the DOOM container is running on the system.

```
root@zarf-test:/home/vagrant# k9s
```

Context: default
Cluster: default
User: default
K9s Rev: v0.24.11 ⚡ v0.24.15
K8s Rev: v1.21.2+k3s1
CPU: 2%
MEM: 25%

<0> all
<1> default
<a> Attach
<ctrl-d> Delete
<d> Describe
<e> Edit
<f> Help
<ctrl-k> Kill

<l> Logs
<p> Logs Previous
<shift-f> Port-Forward
<s> Shell
<f> Show PortForward
<y> YAML

Pods(all) [14]

NAMESPACE+	NAME	PF	READY	RESTARTS	STATUS	CPU	MEM	%CPU/R	%CPU/L	%MEM/R	%MEM/L	IP	NODE	AGE
default	game-c8d98fbb6-c24hw	●	1/1	0	Running	8	8	n/a	n/a	n/a	n/a	10.42.0.16	zarf-test	109s
kube-system	coredns-7448499f4d-qcm1h	●	1/1	0	Running	5	12	5	n/a	18	7	10.42.0.2	zarf-test	7m39s
kube-system	helm-install-traefik-cntnf	●	0/1	1	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.4	zarf-test	7m39s
kube-system	helm-install-traefik-crd-lps6z	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.6	zarf-test	7m39s
kube-system	local-path-provisioner-5ff76fc89d-4wc7d	●	1/1	0	Running	2	7	n/a	n/a	n/a	n/a	10.42.0.3	zarf-test	7m39s
kube-system	metrics-server-86cbb8457f-tnckh	●	1/1	0	Running	2	13	n/a	n/a	n/a	n/a	10.42.0.8	zarf-test	7m39s
kube-system	svclb-traefik-kz47q	●	2/2	0	Running	0	2	n/a	n/a	n/a	n/a	10.42.0.14	zarf-test	7m20s
kube-system	traefik-97b44b794-fkql2	●	1/1	0	Running	2	23	n/a	n/a	n/a	n/a	10.42.0.13	zarf-test	7m20s
logging	helm-install-loki-xz7br	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.5	zarf-test	7m39s
logging	loki-0	●	1/1	0	Running	2	26	n/a	n/a	n/a	n/a	10.42.0.9	zarf-test	7m21s
logging	loki-grafana-c55d7d969-9x4dj	●	1/1	0	Running	1	19	n/a	n/a	n/a	n/a	10.42.0.10	zarf-test	7m21s
logging	loki-promtail-brlt2	●	1/1	0	Running	22	28	n/a	n/a	n/a	n/a	10.42.0.11	zarf-test	7m21s
registry	helm-install-kevin-ddwzg	●	0/1	0	Completed	0	0	n/a	n/a	n/a	n/a	10.42.0.7	zarf-test	7m39s
registry	kevin-docker-registry-8c5bbb65-wnnbg	●	1/1	0	Running	1	17	1	0	3	0	10.42.0.15	zarf-test	7m21s

ZARF has now successfully deployed the PaaS and mission applications. This can be seen with all listed statuses as running or completed. Normal operations with system monitoring and observation using K9s, Promtail, Loki, and Grafana can start.)

Figure 41 shows an image from localhost on the running port. Grafana is a computer resource metrics and observability interface (Grafana Labs, n.d.). Loki in Figure 42 is an infrastructure log aggregation and labeling interface (Grafana Labs Loki, n.d.). The DOOM container (Figure 43) was deployed using ZARF.



Figure 41. Grafana metrics dashboard

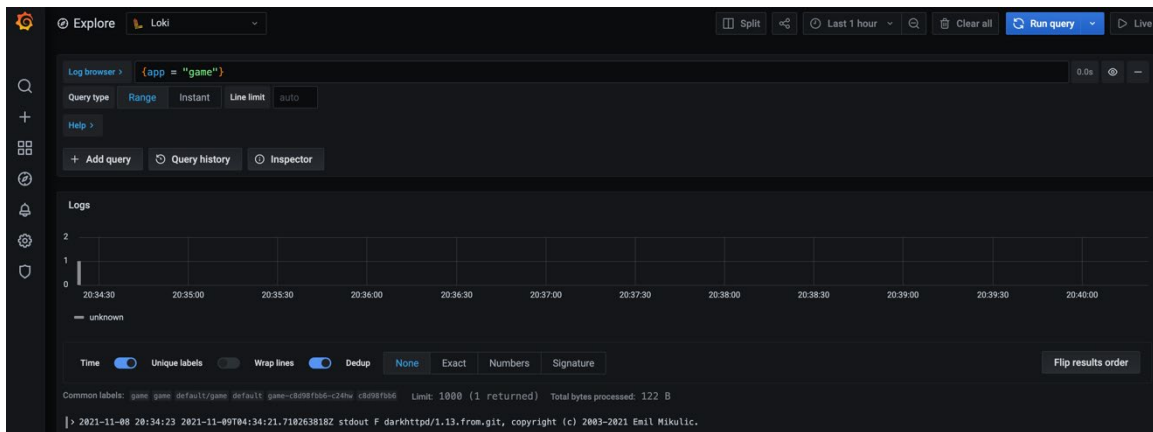


Figure 42. Loki metrics dashboard



Figure 43. DOOM container

ZARF is designed to achieve a declarative state, so installation and uninstallation are easy and quickly repeatable. ZARF destroy requires a --confirm flag to ensure the operator understands that all services and ZARF artifacts will be removed, returning the system to the prior-to-installation condition.

```
root@zarf-test:/home/vagrant# ./zarf destroy --confirm

/usr/local/bin/k3s-remove.sh []
-e
+ [ -s /etc/systemd/system/k3s.service ]
+ basename /etc/systemd/system/k3s.service
+ systemctl stop k3s.service
+ [ -x /etc/init.d/k3s* ]
+ killtree 78979 79072 79688 80593 80609 80800 81205 81389 82502 84332
+ kill -9 78979 79000 79140 79072 79093 79181 79688 79727 80345 80593 80640 81552 80609 80649 82424 80800
80853 82069 81205 81288 81968 81389 81424 81634 81769 82502 82542 82653 84332 84354 84409
+ do_unmount_and_remove /run/k3s
+ awk -v path=/run/k3s $2 ~ ("^" path) { print $2 } /proc/self/mounts
+ sort -r
+ xargs -r -t -n 1 sh -c 'umount "$0" && rm -rf "$0"'
sh -c 'umount "$0" && rm -rf "$0"' /run/k3s/containerd.io.containerd.runtime.v2.task/k8s.io/
ffcc39a2d6e0b611c5eef5d5e8b7ca6765bbcf655f373e9c64571721905811b2/rootfs
```

```
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/ef12aadcf678b1dde9bcea473e0b02300e366d840878272ba2fe26ba433ac1b5/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/e7ff820469df9cf876aebf410917385791640ef8ea3c6a6a3c1d00306c4a33db/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/e530a93822149519c82ac1fb1e8900f42e104d8bf47713739a9e3eba3a4fbb8d/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/d5fecc627d464559643330e7db2ab20f7967ba502c78bcd7dbffda95baf94ba4/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/cd122718baa099b6c734158fc37033aaf3cae2a241e8dc6f6a691ee58f65de95/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/c751eef85bccb5272084fa3a06bf2b7f8671d5926fc270725fed515c353d2d66/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/b5e42584c69b0f3f60ce8c736ae51cae2002955a7ffd4e309adf3687302c4330/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/9e5b1e1639485988a60c0568a8d76bc4348f295c5cbe8918c689ae4307fc71cb/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/9bec7cec9d62de689e22ee44ed05b845feab83cd2f06f48bb4cfa2b51075b057/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/837b89a69532a202f324c30c1c50ac9ef36eacd6845a7433bb6601dfec5f02c5/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/7bb9bfbf8fe3abc4c935385cccd3edf907c77915c8fbc7b90e53399df4883af5/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/71a8945cb1990c4514c978601dbde4c460b48f6c260fe4e018bad81a04f13f56/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/6c4b844c9ff89d3bdc991da6249e2f7bdb20c6e1f9bcb70d9d2411dedbbc26e/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/6c1c0bec46e9d6998dee597b86e9dad6bac0f6df8eabc9afb69a7a2b9ca7f6f1/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/5874cc410d318edb8bc4fbcd259d9b27a86b4e0eefe3d6e84a5130d8433a7190/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/504d8e64e4a5e7ff1ef92905c7dc0bf2a78a41e3cdf1b1fb26b442879f39bb5a/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/4b89d28be8f0b96544886e72fa2e25d5166fa08833448860eb2b52eb3630b3e3/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/1071040f62ff9fb9be7f972e3945578e10bc4fd8d76684efe6e26c787ffd711a/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/094b8a2659c753d643574b4e7b83ad1176074f455d8e7cfb494b6a6d498e0785/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.runtime.v2.task/k8s.io/002b7e54db78e1ac5162188d136c29c5992154ada23adc59b048e5d09117bf0a/rootfs
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/ef12aadcf678b1dde9bcea473e0b02300e366d840878272ba2fe26ba433ac1b5/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/e530a93822149519c82ac1fb1e8900f42e104d8bf47713739a9e3eba3a4fbb8d/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/d5fecc627d464559643330e7db2ab20f7967ba502c78bcd7dbffda95baf94ba4/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/c751eef85bccb5272084fa3a06bf2b7f8671d5926fc270725fed515c353d2d66/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/9e5b1e1639485988a60c0568a8d76bc4348f295c5cbe8918c689ae4307fc71cb/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/9bec7cec9d62de689e22ee44ed05b845feab83cd2f06f48bb4cfa2b51075b057/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/71a8945cb1990c4514c978601dbde4c460b48f6c260fe4e018bad81a04f13f56/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/5874cc410d318edb8bc4fbcd259d9b27a86b4e0eefe3d6e84a5130d8433a7190/shm
sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/4b89d28be8f0b96544886e72fa2e25d5166fa08833448860eb2b52eb3630b3e3/shm
```

```

sh -c 'umount "$0" && rm -rf "$0" /run/k3s/containerd/io.containerd.grpc.v1.cri/sandboxes/
094b8a2659c753d643574b4e7b83ad1176074f455d8e7cfb494b6a6d498e0785/shm
+ do_unmount_and_remove /var/lib/rancher/k3s
+ awk -v path=/var/lib/rancher/k3s $2 ~ ("^" path) { print $2 } /proc/self/mounts
+ xargs -r -t -n 1 sh -c umount "$0" && rm -rf "$0"
+ sort -r
+ do_unmount_and_remove /var/lib/kubelet/pods
+ awk -v path=/var/lib/kubelet/pods $2 ~ ("^" path) { print $2 } /proc/self/mounts
+ + sort -r
xargs -r -t -n 1 sh -c umount "$0" && rm -rf "$0"
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/e8bac361-2a4d-469d-9b83-6188d2b24c56/volume-subpaths/
config/grafana/0
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/e8bac361-2a4d-469d-9b83-6188d2b24c56/volumes/
kubernetes.io~projected/kube-api-access-zjw86
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/d7d0a98e-a3bf-43b4-a9d9-56638f62f44b/volumes/
kubernetes.io~projected/kube-api-access-gb5b8
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/bed3edb2-b6df-4d49-911f-5297c34a2d75/volumes/
kubernetes.io~projected/kube-api-access-2ld4t
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/5b4f6057-9486-47f2-95f3-33a8b5c7f7d5/volumes/
kubernetes.io~projected/kube-api-access-8jcrf
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/4574aaae-aac3-4740-adece-739ca6d1a/volumes/
kubernetes.io~projected/kube-api-access-k9mrz
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/295d90c3-7443-4715-b3da-f08c643f5b64/volumes/
kubernetes.io~projected/kube-api-access-tlrsq
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/28693ab8-734f-4fdb-9c6f-fc4ba477a49e/volumes/
kubernetes.io~projected/kube-api-access-n4sbd
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/270105d6-8b1f-4233-813a-ad1d4520a243/volumes/
kubernetes.io~projected/kube-api-access-vk2qz
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/22cde4a0-9d4e-429e-aa22-a79ec5262ccf/volumes/
kubernetes.io~projected/kube-api-access-nsrd4
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/226134d0-cf38-45c4-a1bd-914003c3c6a2/volumes/
kubernetes.io~secret/config
sh -c 'umount "$0" && rm -rf "$0" /var/lib/kubelet/pods/226134d0-cf38-45c4-a1bd-914003c3c6a2/volumes/
kubernetes.io~projected/kube-api-access-tb8x4
+ do_unmount_and_remove /var/lib/kubelet/plugins
+ awk -v path=/var/lib/kubelet/plugins $2 ~ ("^" path) { print $2 } /proc/self/mounts
+ sort -r
+ xargs -r -t -n 1 sh -c umount "$0" && rm -rf "$0"
+ do_unmount_and_remove /run/netns/cni-
+ awk -v path=/run/netns/cni- $2 ~ ("^" path) { print $2 } /proc/self/mounts
+ xargs -r -t -n 1 sh -c umount "$0" && rm -rf "$0"
+ sort -r
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-df5cac6e-3589-8ce5-3233-2536f1489c1d
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-c88306ec-e967-cc38-cd41-e9d597ea5cf8
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-c004ff6c-a7a3-3d68-9f0f-47ac417e0861
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-b85a731c-22e5-d1e9-1753-f8d121bc2a8c
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-b048034b-80d0-14d0-a221-a9d31776eac4
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-ae5f7d1d-7198-8280-df05-8b8913ed7ab0
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-adb147a8-a926-7692-1766-5d741b4d2805
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-5baa5dfe-6550-66e6-e76d-ab5bef9cfc33
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-165229df-10c0-c4ec-3916-3c71ea4e71bd
sh -c 'umount "$0" && rm -rf "$0" /run/netns/cni-12b88464-dbb5-8786-5823-0b4f31e3ac3f
+ ip netns show
+ xargs -r -t -n 1 ip netns delete
+ grep cni-
+ grep master cni0
+ read ignore iface ignore
+ ip link show

```

```

+ iface=veth0862011f
+ [ -z veth0862011f ]
+ ip link delete veth0862011f
+ read ignore iface ignore
+ iface=vethcbb84fc7
+ [ -z vethcbb84fc7 ]
+ ip link delete vethcbb84fc7
+ read ignore iface ignore
+ iface=veth5f4dc1d1
+ [ -z veth5f4dc1d1 ]
+ ip link delete veth5f4dc1d1
+ read ignore iface ignore
+ iface=vethbfd4e349
+ [ -z vethbfd4e349 ]
+ ip link delete vethbfd4e349
+ read ignore iface ignore
+ iface=veth7aa70140
+ [ -z veth7aa70140 ]
+ ip link delete veth7aa70140
+ read ignore iface ignore
+ iface=veth2aa7692
+ [ -z veth2aa7692 ]
+ ip link delete veth2aa7692
+ read ignore iface ignore
+ iface=vethb1a2a6d5
+ [ -z vethb1a2a6d5 ]
+ ip link delete vethb1a2a6d5
+ read ignore iface ignore
+ iface=veth8c60458d
+ [ -z veth8c60458d ]
+ ip link delete veth8c60458d
+ read ignore iface ignore
+ iface=vetha5e9b28b
+ [ -z vetha5e9b28b ]
+ ip link delete vetha5e9b28b
Cannot find device "vetha5e9b28b"
+ read ignore iface ignore
+ ip link delete cni0
+ ip link delete flannel.1
+ rm -rf /var/lib/cni/
+ iptables-save
+ grep -v KUBE-
+ iptables-restore
+ grep -v CNI-
/usr/bin/systemctl
+ command -v systemctl
+ systemctl disable k3s
Removed /etc/systemd/system/multi-user.target.wants/k3s.service.
+ systemctl reset-failed k3s
+ systemctl daemon-reload
+ rm -f /etc/systemd/system/k3s.service
+ [ -L /usr/local/bin/kubectl ]
+ rm -f /usr/local/bin/kubectl
+ [ -L /usr/local/bin/crictl ]
+ rm -f /usr/local/bin/crictl
+ [ -L /usr/local/bin/ctr ]
+ rm -f /usr/local/bin/ctr
+ rm -rf /etc/rancher/k3s

```

```
+ rm -rf /run/k3s
+ rm -rf /run/flannel
+ rm -rf /var/lib/rancher/k3s
+ rm -rf /var/lib/kubelet
+ rm -f /usr/local/bin/k3s
+ rm -f /usr/local/bin/ctr
+ rm -f /usr/local/bin/crictl
+ rm -f /usr/local/bin/kubectl
+ rm -f /usr/local/bin/k9s
+ rm -f /usr/local/bin/k3s-remove.sh
+ rm -fr zarf-pki
-e
```

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

- Barnett, J. (2021, September 14). *Air Force software platform expansion stalled by cybersecurity concerns*. <https://www.fedscoop.com/air-force-software-platform-expansion-stalled-by-cybersecurity-concerns/>
- Beast Code (n.d). *Ironbank containers / beast code / beast core / base image*. GitLab. Retrieved September 16, 2021, from <https://repo1.dso.mil/dsop/beast-code/beast-core/base-image>
- Black Pearl (n.d). *About Black pearl*. Retrieved February 5, 2021, from <https://blackpearl.us/>
- Bowen, M. (2021, July 6). *U.S. Army Futures Command partners with VMware to create software factory*. *Intelligent CIO North America*. <https://www.intelligentcio.com/north-america/2021/07/06/us-army-futures-command-partners-with-vmware-to-create-software-factory/#>
- Butt, U. J., Abbod, M., Lors, A., Jahankhani, H., Jamal, A., & Kumar, A. (2019, January). Ransomware threat and its impact on SCADA. In *2019 IEEE 12th International Conference on Global Security, Safety and Sustainability (ICGS3)* (pp. 205–212). IEEE.
- Cameron, E., Green M. (2009). Making Sense of Change Management. <http://www.uop.edu.pk/ocontents/Change%20Management%20Book.pdf>
- Chaillan, N. (2020) *How did the Department of Defense move to Kubernetes and Istio?* <https://csrc.nist.gov/CSRC/media/Presentations/dod-enterprise-devsecops-initiative/images-media/DOD%20Enterprise%20DevSecOps%20Initiative%20%20v2.5.pdf>
- Chief of Naval Operations [CNO] (2018, December 26). NAVADMIN 315/18 *Transforming our end-to-end information environment - compile to combat in 24 hours implementation framework*. <https://www.mynavyhr.navy.mil/Portals/55/Messages/NAVADMIN/NAV2018/NAV18315.txt?ver=nQxJ2VXdFMkN8A5sSTFVAg%3D%3D>
- Chief of Naval Operations [CNO] (2020, December 29). NAVADMIN 342/20 *Navy Development Security Operations (DEVSECOPS) guidance*. <https://www.navy.mil/Resources/NAVADMINs/Message/Article/2460115/navy-development-security-operations-devsecops-guidance/no/>
- Chief of Naval Operations [CNO] (2021, January). *CNO NAVPLAN 2021*. <https://media.defense.gov/2021/Jan/11/2002562551/-1/-1/1/CNO%20NAVPLAN%202021%20-%20FINAL.PDF>

- CHIPS Magazine (2018). *Navy Aims for “Compile to Combat in 24 Hours.”* doncio.navy.mil. <https://www.doncio.navy.mil/CHIPS/ArticleDetails.aspx?ID=10501>.
- Commandant of the Marine Corps (2020, January 17). *Marine Corps Order 4141.22 Condition Based Maintenance Plus (CBM+) Order*. <https://www.marines.mil/Portals/1/Publications/MCO%204151.22.pdf>
- Containerd (2021). Containerd An industry-standard container runtime with an emphasis on simplicity, robustness and portability. <https://containerd.io/>.
- Defense Information Systems Agency (2017a, January 31). *DOD Secure Cloud Computing Architecture Functional Requirements*. https://rmf.org/wp-content/uploads/2018/05/SCCA_FRD_v2-9.pdf
- Defense Information Systems Agency (2017b, March 6). *Department of Defense Cloud Computing Security Requirements Guide*. https://rmf.org/wp-content/uploads/2018/05/Cloud_Computing_SRG_v1r3.pdf
- Defense Information Systems Agency (2020, October 15). *Container Hardening Guide*. https://dl.dod.cyber.mil/wp-content/uploads/devsecops/pdf/Final_DevSecOps_Enterprise_Container_Hardening_Guide_1.1.pdf
- Department of Defense Education Activity (2019, October 29) DoDEA Administrative Instruction 8510.01. <https://www.dodea.edu/Offices/PolicyAndLegislation/upload/DoDEA-AI-8510-01-Risk-Management-Framework.pdf>
- Department of Defense [DOD] (2018a, September). Summary Department of Defense Cyber Strategy 2018. https://media.defense.gov/2018/Sep/18/2002041658/-1/-1/1/CYBER_STRATEGY_SUMMARY_FINAL.PDF
- Department of Defense [DOD] (2018b, December). *DOD Cloud Strategy*. <https://media.defense.gov/2019/Feb/04/2002085866/-1/-1/1/DOD-CLOUD-STRATEGY.PDF>.
- Department of Defense [DOD] (2019a, July 12). *DOD DIGITAL MODERNIZATION STRATEGY*. <https://media.defense.gov/2019/Jul/12/2002156622/-1/-1/1/DOD-DIGITAL-MODERNIZATION-STRATEGY-2019.PDF>
- Department of Defense [DOD] (2019b, December). *DOD Product Support manager Guidebook*. <https://www.dau.edu/guidebooks/Shared%20Documents/PSM%20Guidebook.pdf>.

- Department of Defense [DOD] (2020, May 22). Memorandum of Understanding between The Deputy Assistant Secretary of the Navy for Information Warfare and Enterprise Services (DASN IWAR) / Department of the Navy Chief Technology Officer (CTO) AND The Air Force Chief Software Officer (CSO) FOR Black Pearl and Platform One Partnership. https://software.af.mil/wp-content/uploads/2020/12/BlackPearl_P1-MOU_Final_IWAR.pdf
- Department of Defense [DOD] (2021a, March). *DevSecOps Fundamentals Guidebook: DevSecOps Tools & Activities*. Retrieved from <https://software.af.mil/wp-content/uploads/2021/05/DOD-Enterprise-DevSecOps-2.0-Tools-and-Activities-Guidebook.pdf>.
- Department of Defense [DOD] (2021b, March). *DOD Enterprise DevSecOps Strategy Guide*. Retrieved from <https://software.af.mil/wp-content/uploads/2021/05/DOD-Enterprise-DevSecOps-2.0-Strategy-Guide.pdf>.
- Department of Defense [DOD] (2021c, March). *DevSecOps Fundamentals Playbook*. <https://software.af.mil/wp-content/uploads/2021/05/DOD-Enterprise-DevSecOps-2.0-Playbook.pdf>.
- Department of Defense [DOD] (2021d, May 19). DOD Enterprise DevSecOps Strategy Guide. <https://software.af.mil/wp-content/uploads/2021/05/DOD-Enterprise-DevSecOps-2.0-Strategy-Guide.pdf>.
- Department of Defense [DOD] (2021e, July 29). *Department of Defense (DOD) Cloud Native Access Point (CNAP) Reference Design (RD)*. https://dodcio.defense.gov/Portals/0/Documents/Library/CNAP_RefDesign_v1.0.pdf.
- Department of the Navy Chief Information Officer (2020). *Undersecretary of the Navy (UNSECNAV) Memo Operation CATTLE DRIVE*. <https://www.doncio.navy.mil/ContentView.aspx?id=14182>.
- Department of the Navy Chief Information Officer (2021, September 14). *Department of the Navy and Department of the Air Force Cybersecurity Reciprocity*. <https://www.doncio.navy.mil/contentview.aspx?id=15100>.
- Derailed (n.d.). *Derailed/k9s: Kubernetes CLI to manage your clusters in style!* GitHub. Retrieved July 19, 2021, from <https://github.com/derailed/k9s>.
- DeRosa, D. (2021, May 12). *U.S. Navy selects Lockheed Martin and IFS to deliver intelligent ship and aircraft maintenance: IFS France*. <https://www.ifs.com/fr/news-and-events/newsroom/2021/05/12/us-navy-selects-lockheed-martin-and-ifs-to-deliver-intelligent-ship-and-aircraft-maintenance/>.
- Docker (n.d. a). *Empowering app development for developers*. Retrieved July 30, 2021, from <https://www.docker.com/>.

- Docker (n.d. b). *What is a container?* Retrieved July 30, 2021, from <https://www.docker.com/resources/what-container>.
- Eckstein, M. (2020, August 3). *Sub base kings bay keeping current Ohio subs ready, prepping for incoming Columbia class*. USNI News. <https://news.usni.org/2020/08/03/sub-base-kings-bay-keeping-current-ohio-subs-ready-prepping-for-incoming-columbia-class>.
- Elastic (n.d.). *Infrastructure monitoring with Elasticsearch*. Retrieved August 10, 2021, from <https://www.elastic.co/infrastructure-monitoring>.
- Eversden, A. (2021, September 2). *Air Force chief software officer knocks DOD as he departs*. <https://www.c4isrnet.com/battlefield-tech/it-networks/2021/09/02/air-force-chief-software-offer-knocks-dod-as-he-departs/>.
- Gamboa, E. (2021, August 17). *NAVWAR deploys the Navy's first application arsenal*. <https://www.dvidshub.net/news/403380/navwar-deploys-navys-first-application-arsenal>.
- Garamone, J. (2020, November 30). *Joint All-Domain Command, Control Framework belongs to warfighters*. <https://www.defense.gov/News/News-Stories/Article/Article/2427998/joint-all-domain-command-control-framework-belongs-to-warfighters/>.
- Gitea (n.d.). *A painless self-hosted Git service*. Retrieved July 16, 2021, from <https://gitea.io/en-us/>.
- Github (n.d.). *Defenseunicorns/zarf: K8s Airgap buddy*. Retrieved November 5, 2021, from <https://github.com/defenseunicorns/zarf>.
- GitLab (n.d.). *What is Gitops?* Retrieved May 10, 2021, from <https://about.gitlab.com/topics/gitops/>.
- GO (n.d.). *Build fast, reliable, and efficient software at scale*. Retrieved March 8, 2021, from <https://go.dev/>.
- Google Cloud (n.d. a). *DevOps Measurement: Monitoring and Observability*. Retrieved August 12, 2021, from <https://cloud.google.com/architecture/devops/devops-measurement-monitoring-and-observability>
- Google Cloud (n.d. b). *Explore Dora's research program*. Retrieved August 12, 2021, from <https://www.devops-research.com/research.html>.
- Government Accountability Office (2018). *United States Government Accountability Office Report to the Committee on Armed Services, U.S. Senate WEAPONS SYSTEMS CYBERSECURITY DOD Just Beginning to Grapple with Scale and Vulnerabilities (GAO-19-128)*. <https://www.gao.gov/assets/gao-19-128.pdf>.

- Government Accountability Office (2020a). *Navy and Marine Corps: Services Continue Efforts to Rebuild Readiness, but Recovery Will Take Years and Sustained Management Attention* (GAO-21-225t). <https://www.gao.gov/assets/gao-21-225t.pdf>.
- Government Accountability Office (2020b). *Navy Shipyards: Actions Needed to Address the Main Factors Causing Maintenance Delays for Aircraft Carriers and Submarines* (GAO-20-588). <https://www.gao.gov/assets/gao-20-588.pdf>
- Grafana Labs Loki (n.d.). *Grafana Loki*. Retrieved October 12, 2021, from <https://grafana.com/oss/loki/>.
- Grafana Labs Promtail (n.d.). *Promtail documentation*. Retrieved October 12, 2021, from <https://grafana.com/docs/loki/latest/clients/promtail/>.
- Grafana Labs (n.d.). *Grafana features*. Retrieved October 12, 2021, from <https://grafana.com/grafana/>.
- Hiza, G. W. (1982). TRIDENT submarine logistics data system (LDS): a case study in life cycle management and budgeting. Naval Postgraduate School Thesis. <https://calhoun.nps.edu/bitstream/handle/10945/20381/tridentsubmarine00hiza.pdf?sequence=1&isAllowed=y>.
- Howard, T. (2017, May 23). The Navy's software frontier. *The Maritime Executive*. <https://www.maritime-executive.com/blog/the-navys-software-frontier>.
- IBM (n.d.). *What is DevSecOps*. Retrieved October 5, 2021, from <https://www.ibm.com/cloud/learn/devsecops>.
- Istio (2021). *The Istio Service Mesh*. <https://istio.io/latest/about/service-mesh/>.
- Jaeger (2021). *Open source, end-to-end distributed tracing*. <https://www.jaegertracing.io/>.
- K9s (2020). *Who let the pods out?* <https://k9scli.io/>.
- Kiali (n.d.). *The Console for Istio Service Mesh*. Retrieved September 28, 2021, from <https://kiali.io/>.
- Korda, M. (2021). *Alternatives to the Ground Based Strategic Deterrent*. Federation of American Scientists <https://uploads.fas.org/2021/02/Alternatives-to-the-GBSD-Feb.-2021.pdf>.
- Krazit, T. (2021, March 25). *How the U.S. Air Force deployed Kubernetes and Istio on an F-16 in 45 days*. <https://thenewstack.io/how-the-u-s-air-force-deployed-kubernetes-and-istio-on-an-f-16-in-45-days/>.

- Kubernetes (2021). *Kubernetes Documentation*. <https://kubernetes.io/docs/home/>.
- Larson, M. (2014). *Microservices*. <https://aws.amazon.com/microservices/>.
- Macvittie, D. (2020, August 31). *The pros and cons of configuration-as-code*. <https://devops.com/the-pros-and-cons-of-configuration-as-code/>.
- McQuade, J. M., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019, March 14). *Software is Never Done: Refactoring the Acquisition Code for Competitive Advantage v3.3*. https://media.defense.gov/2019/mar/26/2002105908/-1/-1/0/swap.report_supplementary.docs.3.21.19.pdf.
- Mell, P., & Grance, T. (2011). *The NIST definition of cloud computing*. <http://faculty.winthrop.edu/domanm/csci411/Handouts/NIST.pdf>.
- Microsoft Azure (2021a). *What is DevOps? DevOps explained: Microsoft Azure*. <https://azure.microsoft.com/en-us/overview/what-is-devops/#devops-overview>.
- Microsoft Azure (2021b). *What is PaaS? Platform as a service*. <https://azure.microsoft.com/en-us/overview/what-is-paas/>
- National Institute of Standards and Technology (2018, April 16). *Framework for Improving Critical Infrastructure Cybersecurity*. <https://nvlpubs.nist.gov/nistpubs/cswp/nist.cswp.04162018.pdf>
- Naval Information Warfare Center (2018, December). *Risk Management Framework (RMF) Rapid Assess and Incorporate Software Engineering (RAISE) Playbook*. NIWC.
- Naval Information Warfare Center (2020, November 19). *Software Version Description (SVD) Agile Core Services (ACS) 3.3 Component of Consolidated Afloat Networks and Enterprise Services (CANES)*. NIWC
- Nosis (2021, September 7). *Program Overview Brief*.
- Patel, A. (2021, June 30). *U.S. Army Futures Command embraces the power of VMware Tanzu*. <https://tanzu.vmware.com/content/blog/us-army-futures-command-vmware-tanzu>.
- Platform One Big Bang (n.d.) *Platform one / big bang / bigbang*. GitLab. Retrieved January 17, 2021, from <https://repo1.dso.mil/platform-one/big-bang/bigbang>.
- Platform One Party Bus (n.d.). *The environment and services you need to develop and deploy your software application*. Retrieved January 17, 2021, from <https://p1.dso.mil/#/products/party-bus>

- Platform One (n.d.). *Platform One products and services: Iron Bank*. Retrieved January 17, 2021, from <https://software.af.mil/dsop/services/>.
- PMW-160 (2019, May 20). *CANES ROADMAP*.
- PMW-160 (2020, February 14). *CANES Developer's Interface Guide (DIG)*.
- Pope, C. (2020, September 3). *Advanced Battle Management System Field Test brings joint force together across all domains during second onramp*. <https://www.af.mil/News/Article-Display/Article/2336618/advanced-battle-management-system-field-test-brings-joint-force-together-across/>.
- Prodan, S. (2021, March 22). *Podinfo/readme.md at master Stefanprodan/podinfo*. GitHub. <https://github.com/stefanprodan/podinfo/blob/master/README.md>.
- Rancher Labs (n.d.). *K3s - Lightweight Kubernetes*. Retrieved June 16, 2021, from <https://rancher.com/docs/k3s/latest/en/>.
- RedHat (2019, December 2). *What is container orchestration?* <https://www.redhat.com/en/topics/containers/what-is-container-orchestration>.
- RedHat (n.d.) *Aggregating container logs. Aggregating Container Logs | Installation and Configuration | OpenShift Enterprise 3.1*. Retrieved October 19, 2021, from https://docs.openshift.com/enterprise/3.1/install_config/aggregate_logging.html.
- Repo1 Sync (n.d.). *Synker project*. Retrieved May 7, 2021, from <https://repo1.dso.mil/platform-one/delivery/hagrid/sync>.
- Rosenburg, Barry. (2019, June 20). *Fail Fast, Not Twice: DOD's push for Agile software development*. <https://breakingdefense.com/2019/06/fail-fast-not-twice-dods-push-for-agile-software-development/>.
- Schafer, B., & Baker, B. (2019, May 15). *COLUMBIA Class Design for Sustainment Product Support Manger Workshop*. https://www.acq.osd.mil/log/MR/PSM_workshop.html/2019_Files/Day_One/8_Influence_Design_from_Sustainment_Perspective_Baker.pdf
- Schwartz, M. (2017). *A Seat at the Table and The Art of Business Value*. IT Revolution.
- SEA06L (2019, July 18). *Model Based Product Support (MBPS) Overview brief to NSRP - In-service navy PLM Project Kick-off*. https://www.nsrp.org/wp-content/uploads/2019/08/05-MBPS_Overivew_June-2019-Updated_v5.pdf.
- Shelbourne, M. (2020, October 29). *Navy's 'Project Overmatch' structure aims to accelerate creating naval battle network*. <https://news.usni.org/2020/10/29/navys-project-overmatch-structure-aims-to-accelerate-creating-naval-battle-network>.

- Shirey, R. (2007, August). *rfc4949 Internet Security glossary*. Document search and retrieval page. <https://datatracker.ietf.org/doc/html/rfc4949>.
- Sirota, S. (2021, April 2). *U-2 Dragon LADY executes FIRST in-flight test OF DevSecOps capabilities*. <https://insidedefense.com/insider/u-2-dragon-lady-executes-first-flight-test-devsecops-capabilities>.
- Smith, B., Castelle, K., Slaughter, R., & Bradley, J. (2021). *The GitOps Hypothesis: Exploration of Sustainment Benefits of Declarative GitOps Software Deployment Models for Navy Platforms*. [Manuscript submitted for publication] American Society of Naval Engineers.
- Souppaya, M., Morello, J., & Scarfone, K. (2017, September). *NIST Special Publication 800–190 Application Container Security Guide*. <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-190.pdf> nvlpubs.nist.gov.
- Space and Naval Warfare Systems Command (2011, November). *Consolidated Afloat Network and Enterprise Services Fact Sheet*. <https://www.secnaw.navy.mil/rda/Documents/canes+overview+for+asn+rda+11-2-11-s.pdf>
- Terraform (n.d.). *Write, plan, apply*. Retrieved November 1, 2021, from <https://www.terraform.io/>.
- Underwood, K. (2020, March 3). *U.S. Navy builds flexibility into its digital transformation*. <https://www.afcea.org/content/us-navy-builds-flexibility-its-digital-transformation>.
- White House (2021, July 28). *National Security Memorandum on Improving Cybersecurity for Critical Infrastructure Control Systems*. <https://www.whitehouse.gov/briefing-room/statements-releases/2021/07/28/national-security-memorandum-on-improving-cybersecurity-for-critical-infrastructure-control-systems/>
- Wilson, B., Riposo, J., Goughnour, T., Burns, R. M., Vermeer, M. J., Kochhar, A. K., ... & Eisman, M. (2020). *Naval Aviation Maintenance System: Analysis of Alternatives*. RAND. <https://apps.dtic.mil/sti/pdfs/AD1095540.pdf>.
- Ziezulewicz, G. (2021, August 12). *Audit: Some Navy sub cybersecurity inspections were neglected in recent years*. <https://www.defensenews.com/news/your-navy/2021/08/12/audit-some-navy-sub-cybersecurity-inspections-were-neglected-in-recent-years/?contentQuery=%7B%22section%22%3A%22%2Fnaval%22%2C%22from%22%3A38%2C%22size%22%3A10%2C%22exclude%22%3A%22%22%7D&contentFeatureId=f0fljrziYHYB5pv>.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California