CLEARED
For Open Publication

Oct 19, 2021

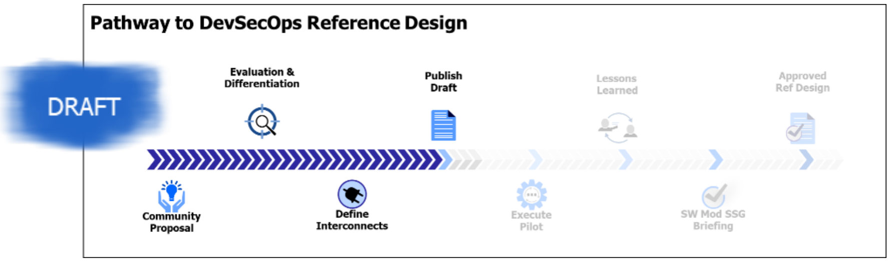
Department of Defense
OFFICE OF PREPUBLICATION AND SECURITY REVIEW

DoD Enterprise DevSecOps Reference Design:

AWS Managed Services (DoD IaC Baseline)

September 2021

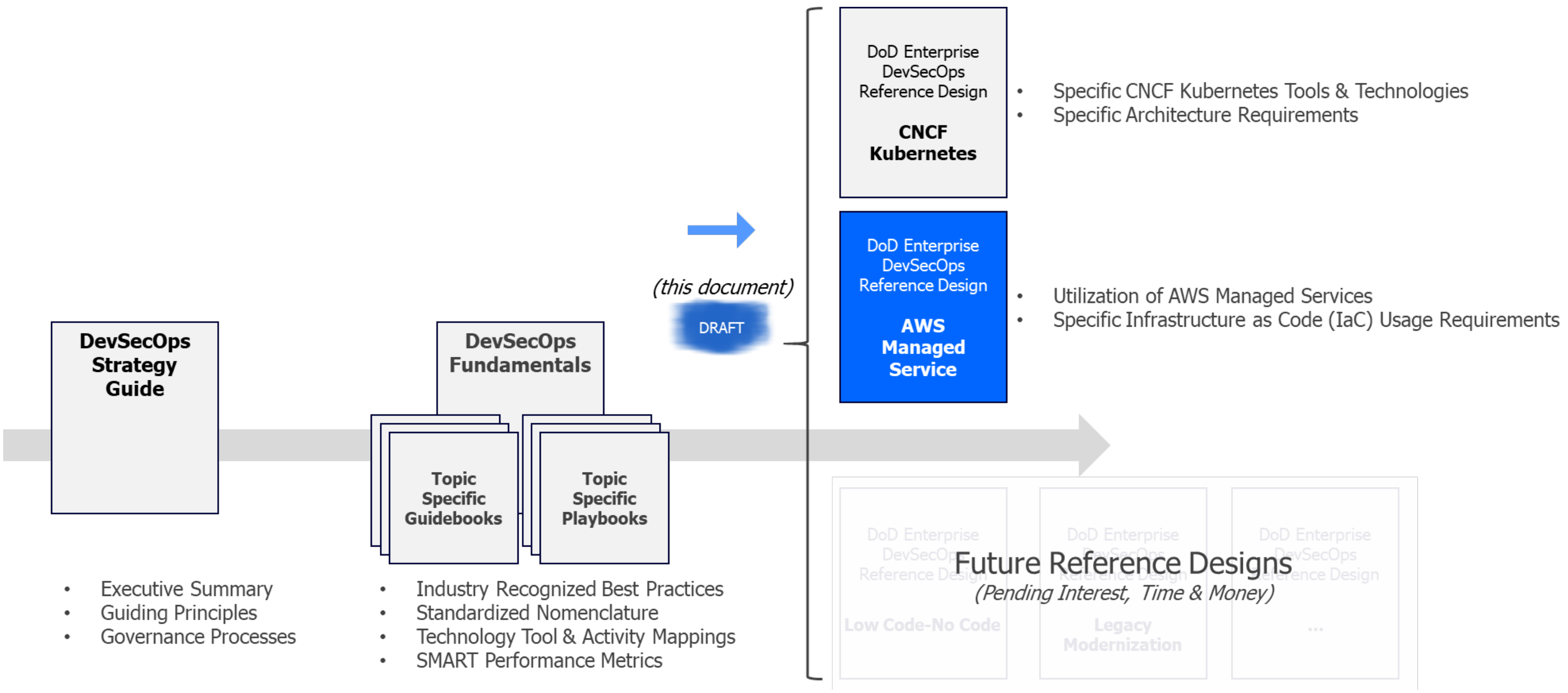
Version 0.2



This document automatically expires 1-year from publication date unless revised.

DISTRIBUTION STATEMENT A. Approved for public release. Distribution is unlimited.

Document Set Reference



DevSecOps Strategy Guide

- Executive Summary
- Guiding Principles
- Governance Processes

DevSecOps Fundamentals

Topic Specific Guidebooks

Topic Specific Playbooks

- Industry Recognized Best Practices
- Standardized Nomenclature
- Technology Tool & Activity Mappings
- SMART Performance Metrics

(this document)

DRAFT

DoD Enterprise DevSecOps Reference Design

CNCF Kubernetes

- Specific CNCF Kubernetes Tools & Technologies
- Specific Architecture Requirements

DoD Enterprise DevSecOps Reference Design

AWS Managed Service

- Utilization of AWS Managed Services
- Specific Infrastructure as Code (IaC) Usage Requirements

Future Reference Designs

(Pending Interest, Time & Money)

DoD Enterprise DevSecOps Reference Design

Low Code-No Code

DoD Enterprise DevSecOps Reference Design

Legacy Modernization

DoD Enterprise DevSecOps Reference Design

…

Document Approvals

Approved by:

TBD

Trademark Information

Names, products, and services referenced within this document may be the trade names, trademarks, or service marks of their respective owners. References to commercial vendors and their products or services are provided strictly as a convenience to our readers, and do not constitute or imply endorsement by the Department of any non-Federal entity, event, product, service, or enterprise.

Contents

Figures

Tables

1 Introduction

1.1 Background

Modern information systems and weapons platforms are driven by software. As such, the DoD is working to modernize its software practices to provide the agility to *deliver resilient software at the speed of relevance*. DoD Enterprise DevSecOps Reference Designs are expected to provide clear guidance on how specific collections of technologies come together to form a secure and effective software factory.

1.2 Purpose

This DoD Enterprise DevSecOps Reference Design is specifically for a collection of Amazon Web Services (AWS) managed services. The managed services explicitly identified as part of this reference design are built from Infrastructure as Code (IaC) baselines that leverage automation to generate preconfigured, preauthorized, Platform as a Service (PaaS) focused environments. These environments, whenever possible, leverage security services offered by the Cloud Service Provider (CSP, AWS in this case) over traditional datacenter tools.

A Cloud Native Computing Foundation (CNCF) Certified Kubernetes implementation remains central to this reference design, offering an elastic instantiation of a DevSecOps factory in the specific CSP. It provides a formal description of the key design components and processes to provide a repeatable reference design that can be used to instantiate a DoD DevSecOps Software Factory powered by Kubernetes. This reference design is aligned to the DoD Enterprise DevSecOps Strategy, and aligns with the baseline nomenclature, tools, and activities defined in the DevSecOps Fundamentals document and its supporting guidebooks and playbooks.

Adoptees of this reference design must recognize and understand that there is a certain degree of vendor lock-in that occurs when leveraging the security services offered by the CSP. Additional lock-in may occur if teams utilize proprietary features unique to the CSP.

For brevity, the use of the term 'Kubernetes' or 'K8s' throughout the remainder of this document must be interpreted as a Kubernetes implementation that properly submitted software conformance testing results to the CNCF for review and corresponding certification. The CNCF lists over 90 Certified Kubernetes offerings that meet software conformation expectations. [1]

[1] Cloud Native Computing Foundation, “Software conformance (Certified Kubernetes,” [ONLINE] Available: https://www.cncf.io/certification/software-conformance/. [Accessed 8 February 2021].

The target audiences for this document include:

- DoD Enterprise DevSecOps capability providers who build DoD Enterprise DevSecOps hardened containers and provide a DevSecOps hardened container access service.
- DoD Enterprise DevSecOps capability providers who build DoD Enterprise DevSecOps platforms and platform baselines and provide a DevSecOps platform service.
- DoD organization DevSecOps teams who manage (instantiate and maintain) DevSecOps software factories and associated pipelines for its programs.
- DoD program application teams who use DevSecOps software factories to develop, secure, and operate mission applications.
- Authorizing Officials (AOs).

This reference design aligns with these reference documents:

- DoD Digital Modernization Strategy.[2]
- DoD Cloud Computing Strategy.[3]
- DISA Cloud Computing Security Requirements Guide.[4]
- DISA Secure Cloud Computing Architecture (SCCA).[5]
- Presidential Executive Order on Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure (Executive Order (EO) 1380).[6]
- National Institute of Standards and Technology (NIST) Cybersecurity Framework.[7]
- NIST Application Container Security Guide.[8]
- Kubernetes STIG.[9]
- DISA Container Hardening Process Guide.[10]

[2] DoD CIO, *DoD Digital Modernization Strategy,* Pentagon: Department of Defense, 2019.

[3] Department of Defense, "DoD Cloud Computing Strategy," December 2018.

[4] Defense Information Systems Agency, "Department of Defense Cloud Computing Security Requirements Guide, v1r3," March 6, 2017

[5] Defense Information Systems Agency, "DoD Secure Cloud Computing Architecture (SCCA) Functional Requirements," January 31, 2017.

[6] White House, "Presidential Executive Order on Strengthening the Cybersecurity of Federal Networks and Critical Infrastructure (EO 1380)," May 11, 2017.

[7] National Institute of Standards and Technology, Framework for Improving Critical Infrastructure Cybersecurity, 2018.

[8] NIST, "NIST Special Publication 800-190, Application Container Security Guide," September 2017.

[9] Defense Information Systems Agency, "Kubernetes STIG, Version 1, Release 2," July 26, 2021.

[10] Defense Information Systems Agency, "Container Hardening Process Guide, V1R1," October 15, 2020

- Cloud Native Access Point Reference Design.[11]

1.3 DevSecOps Compatibility

This reference design asserts version compatibility with these supporting DevSecOps documents:

- DoD Enterprise DevSecOps Strategy Guide, Version 2.1.
- DevSecOps Tools and Activities Guidebook, Version 2.1.

1.4 Scope

This reference design is not vendor or product-agnostic. It is built upon the managed services offered by AWS as a CSP and may lead to a degree of vendor and/or managed service lock-in. This reference design provides specific execution guidance for use by software teams. It is applicable to developing new capabilities and to sustaining existing capabilities in both business and weapons systems software, including business transactions, C3, embedded systems, big data, and Artificial Intelligence (AI).

This document does not address strategy, policy, or acquisition.

1.5 Document Overview

The documentation is organized as follows:

- Section 1 describes the background, purpose and scope of this document.
- Section 2 identifies the assumptions relating to this design.
- Section 3 describes the DevSecOps software factory interconnects unique to a Kubernetes reference built from IaC and a specific set of AWS managed services.
- Section 4 describes the containerized software factory design.
- Section 5 captures the additional required and preferred tools and activities, building upon the DevSecOps Tools and Activities Guidebook as a baseline.

[11] DoD CIO, "DoD Cloud Native Access Point Reference Design, v1.0" [Online] Available at: https://dodcio.defense.gov/Portals/0/Documents/Library/CNAP_RefDesign_v1.0.pdf

2 Assumptions and Principles

This reference design makes the following assumptions:

- This reference design will remain in a Pre-Decisional/DRAFT status until all requisite managed services have been approved for use. Specifically, AWS App Mesh is presently under 3PAO Assessment.[12]
- The AWS Kubernetes managed service, *Elastic Kubernetes Service (EKS)* is utilized. This managed service has submitted conformance testing results for review and certification by the CNCF.
- **Vendor lock-in at the CSP level occurs with this reference design**, amplified further if a development team takes advantage of CSP specific features within the K8s managed service.
- Product lock-in into the Kubernetes API and its overall ecosystem is openly recognized.

It is critically important to avoid the proprietary APIs that are sometimes added by vendors on top of the existing CNCF Kubernetes APIs. These APIs are not portable and may create vendor lock-in at the application level, not just at the software factory level!

- Adoption of hardened containers as a form of immutable infrastructure results in standardization of common infrastructure components that achieve consistent and predictable results.
- This reference design may depend upon DoD Enterprise Services, which will be named when applicable.

2.1 Benefits of Adopting DoD Cloud Infrastructure as Code (IaC)

Software teams adopting this reference design can realize the following benefits from the IaC scripts:

- Deep CSP service and platform adoption (e.g. platform as a service (PaaS), software as a service (SaaS), serverless) maximizes the benefits of public cloud adoption. This is in contrast to the Cloud agnostic approach captured in the separate DevSecOps Reference Design for CNCF Kubernetes.
- Increased speed to market by consciously choosing “buy” versus “build”
- Segmented Environments

[12] Amazon Web Services, “AWS Services in Scope by Compliance Program, DoD CC SRG,” [Online] Available at: https://aws.amazon.com/compliance/services-in-scope/.

- Enforcement of centralized logging of both services and applications
- Continuous monitoring accessible via a dashboard interface
- Elimination of high-risk management plane protocols, including SSH and RDP
- Enforcement of only HTTPS traffic into the environment
- Isolation and protection of web applications via an Application Gateway and a Web Application Firewall (WAF)
- Automatic change management using Infrastructure as Code via the AWS Cloud Development Kit and AWS CloudFormation.
- Elimination of manual deployable object administration using a continuous integration/continuous delivery (CI/CD) pipeline in conjunction with the main branch of a repository.
- All hardening and patching of the PaaS is the responsibility of the CSP
- DoD Cloud IaC includes prebuilt integration with DISA's Global Directory, depicted in Figure 1, to provide DoD-wide CAC-enabled application authentication. Global Directory is the new enterprise identification and authentication service from DISA's Cloud Computing Program Office (CCPO) and it received an ATO in November, 2020. Further, Global Directory provides unprecedented interoperability across all impact level 5 (IL-5) tenants and applications, permitting access to multiple environments with a singular user account.

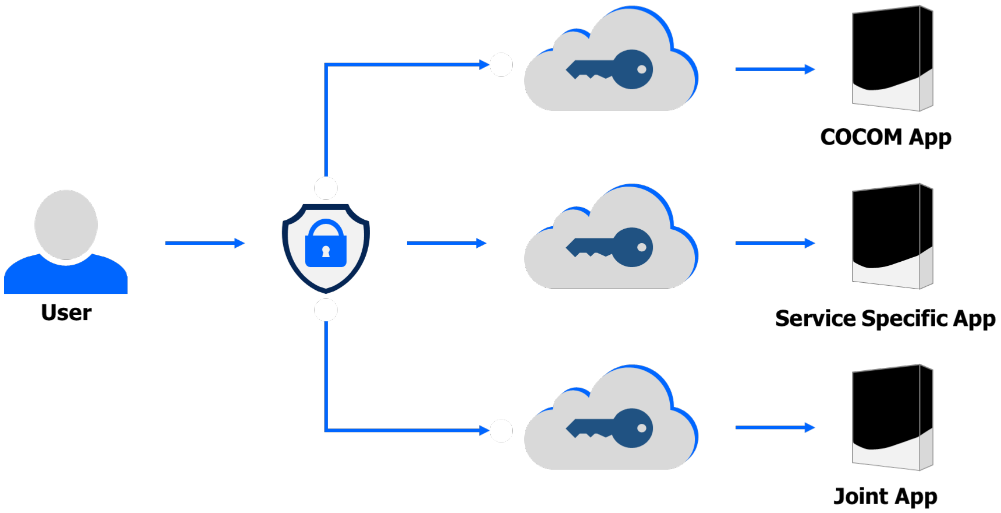


Figure 1: Visualization of the benefits of using DISA's Global Directory

3 Software Factory Interconnects

The DevSecOps Fundamentals describes a DevSecOps platform as a multi-tenant environment consisting of three distinct layers: *Infrastructure*, *Platform/Software Factory*, and *Application(s)*. Each reference design is expected to identify its unique set of tools and activities that exist within and/or at the boundaries between the discrete layers. These unique tool sets or configurations are known as *Reference Design Interconnects*. Well-defined interconnects in a reference design enable tailoring of the software factory design, while ensuring that core capabilities of the software factory remain intact.

Figure 2: AWS Managed Services Reference Design Interconnects identifies the specific managed services present created at the end of IaC execution in the AWS cloud.

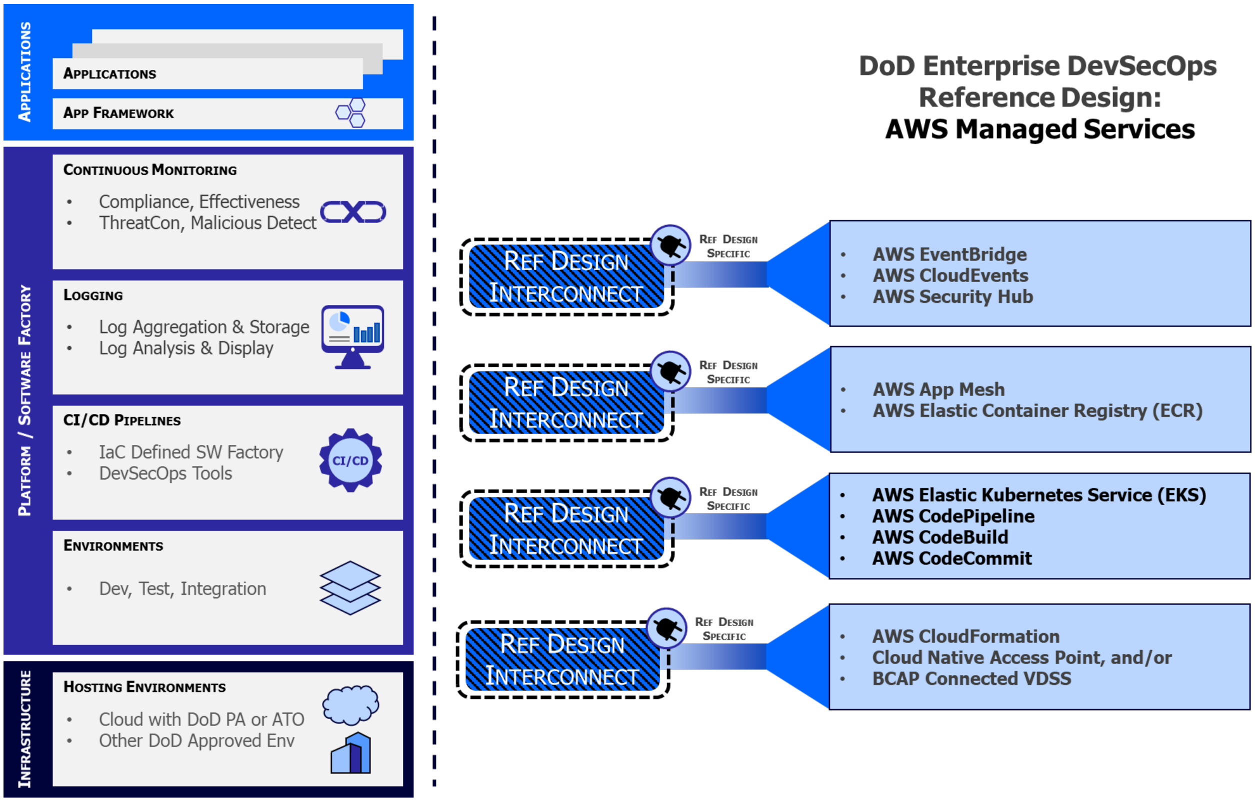


Figure 2: AWS Managed Services Reference Design Interconnects

Highlighted interconnects include:

- Cloud Native Access Point (CNAP) is the Enterprise Architecture Engineering Panel and Digital Modernization Infrastructure, Executive Committee (DMI EXCOM) approved enterprise architecture at the infrastructure layer to manage all north-south network traffic.

- Use of AWS managed Kubernetes service (EKS) in each environment.
- Use of AWS managed container registry (ECR) as a locally centralized artifacts repository to host hardened containers from Iron Bank, the DoD Centralized Artifact Repository (DCAR) of hardened and centrally accredited software containers.
- Use of AWS App Mesh within the EKS orchestrator to manage all east-west network traffic using Zero Trust (ZT) principles down to the container/function level.
- Use of AWS CloudEvents, EventBridge, and Security Hub for continuous monitoring.

Each of these interconnects will be described fully next.

3.1 Cloud Native Access Points

A Cloud Native Access Point (CNAP) provides a zero-trust architecture to provide access to development, testing, and production enclaves at Impact Level 2 (IL-2), Impact Level 4 (IL-4), and Impact Level 5 (IL-5).[13] CNAP provides access to the DevSecOps environments by using an internet-facing Cloud-native zero trust environment. CNAP's zero trust architecture facilitates development team collaboration from disparate organizations.

3.2 CNCF Certified Kubernetes: AWS Elastic Kubernetes Service

Kubernetes is a *container orchestrator* that manages the scheduling and execution of Open Container Initiative (OCI) compliant containers across multiple nodes, depicted in *Figure 3*. OCI is an open governance structure for creating open industry standards around both container formats and runtimes.[14] The container is the standard unit of work in this reference design. Containers enable software production automation in this reference design, and they also allow operations and security process orchestration.

[13] Defense Information Systems Agency, "Department of Defense Cloud Computing Security Requirements Guide, v1r3," Mar 6, 2017

[14] The Linux Foundation Projects, "Open Container Initiative," [Online] Available at: https://opencontainers.org.

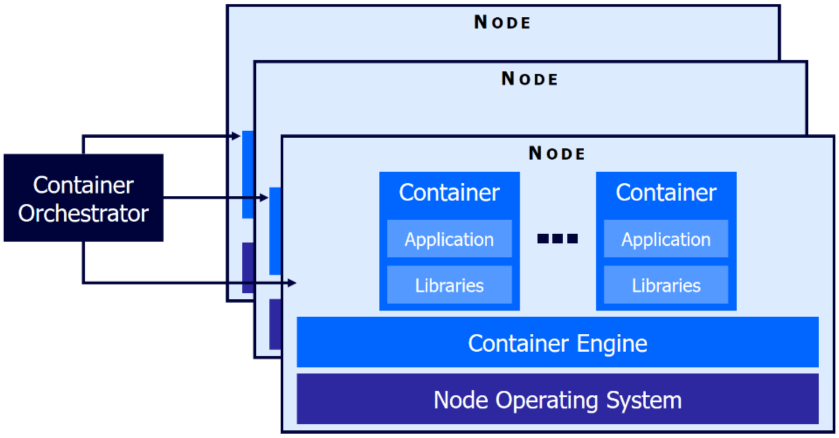


Figure 3: Container Orchestrator and Notional Nodes

Kubernetes provides an API that ensures total abstraction of orchestration, compute, storage, networking, and other core services that guarantees software can run in any environment, from the Cloud to embedded inside of platforms like jets or satellites.

The key benefits of adopting AWS EKS include:

- **Managed Service Environment**: As a managed service, AWS manages the availability and scalability of the Kubernetes control plane nodes responsible for scheduling containers, managing the availability of applications, storing cluster data, and other pertinent tasks.
- **Deployment Flexibility**: AWS EKS supports running a K8s application on both Amazon Elastic Computing Cloud (EC2) and Fargate, a service that offers serverless compute for software containers.
- **Service Integrations:** AWS Controllers for Kubernetes (ACK) provides support to directly manage AWS services from Kubernetes, further simplifying scalable
- **Hosted Kubernetes Console:** EKS provides developers and operations personnel an integrated console for organizaing, visualizing, and troubleshooting applications deployed on the Kubernetes cluster.
- **Baked-In Security**: A sidecar, such as the Sidecar Container Security Stack, is automatically injected into any K8s cluster with zero trust.
- **Resiliency:** Self-healing of unstable or crashed containers.
- **Adaptability:** Containerized microservices create highly-composable ecosystems.
- **Automation**: Fundamental support for a GitOps model and IaC speed processes and feedback loops.
- **Scalability**: Application elasticity to appropriately scale and match service demand.

The adoption of K8s and OCI compliant containers are concrete steps towards true microservice reuse, providing the Department with a compelling ability to pursue higher orders of code reuse across an array of programs.

3.3 Locally Centralized Artifact Repository: AWS Elastic Container Registry

A Locally Centralized Artifact Repository is a local repository tied to the software factory. It stores artifacts pulled from Iron Bank, the DoD repository of digitally signed binary container images that have been hardened. The local artifact repository also likely stores locally developed artifacts used in the DevSecOps processes. Artifacts stored here include, but are not limited to, container images, binary executables, virtual machine (VM) images, archives, and documentation.

The Iron Bank artifact repository provides hardened, secure technical implementation guide (STIG) compliant, and centrally updated, scanned, and signed containers that increases the cyber survivability of these software artifacts. At time of writing this reference design, over 800 artifacts were in Iron Bank, with more being added continuously.

Programs may opt for a single artifact repository and rely on the use of tags to distinguish between the different content types. It is also permissible to have separate AWS ECR artifact repositories to store local artifacts and released artifacts.\

3.4 Incorporate Zero Trust Principles: AWS App Mesh

The cyber arena is an unforgiving hostile environment where even a minute exposure and compromise can lead to catastrophic failures and loss of human life. Industry norms now recognize that a modern holistic cybersecurity posture must include centralized logging and telemetry, zero trust ingress/egress/east-west network traffic, and behavior detection at a *minimum*.

AWS App Mesh delivers appropriate end-to-end visibility by capturing metrics, logs, and traces all deployed application components. Beyond visibility, AWS App Mesh also encrypts all network traffic between services even when they are within a private networking enclave. When these capabilities are combined with Amazon CloudWatch and other cybersecurity tools for monitoring and tracking, rapid issue isolation is possible.

The highlights of AWS App Mesh include:

- Traffic routing to declaratively connect services.
- Client-side traffic policies to create resiliency.
- Zero Trust security model.
- Mutual Transport Layer Security (mTLS) enforces service-to-service identify verification.

Table 1: AWS Managed Service Cybersecurity Aspects

Tool	Features	Benefits	Baseline
AWS App Mesh Proxy	Proxy automatically checks-in with and is configured by the App Mesh managed service.	Declarative traffic routing, mandatory mTLS, and client-side traffic policy enforcement.	REQUIRED
Logging Storage and Retrieval Service	Stores logs and allows searching logs	Place to store logs	REQUIRED
Log visualization and analysis	Ability to visualize log data in various ways and perform basic log analysis.	Helps to find anomalous patterns	PREFERRED
Container policy enforcement	Support for Security Content Automation Protocol (SCAP) and container configuration policies. These policies can be defined as needed.	Automated policy enforcement	REQUIRED
Runtime Defense	Creates runtime behavior models, including whitelist and least privilege	Dynamic, adaptive cybersecurity	REQUIRED
Vulnerability Management	Provides vulnerability management	Makes sure everything is properly patched to avoid known vulnerabilities	REQUIRED
Artifact Repository	Storage and retrieval for artifacts such as containers.	One location to obtain hardened artifacts such as containers	REQUIRED
Zero Trust model down to the container level	Provides strong identities per Pod with certificates, mTLS tunneling and whitelisting of East-West traffic down to the Pod level.	Reduces attack surface and improves baked-in security	REQUIRED

4 Software Factory K8s Reference Design

Adoption of a managed service still requires automation, environmental drift control, and adoption of IaC. To support a CSP managed service DevSecOps reference design, utilization of DoD Cloud IaC scripts to generate **pre-configured, pre-authorized, Platform as a Service (PaaS)** software factory environments is expected. In addition to the use of AWS CSP managed services, DoD Cloud IaC, adoptees are expected to use hardened software container from Iron Bank to create the underlying software factory DevSecOps tooling components.

All software factory implementations follow the DevSecOps philosophy and go through four unique phases: Design, Instantiate, Verify, Operate & Monitor. *Figure 4* illustrates the phases, activities, and the relationships with the application lifecycle. Security is applied across all software factory phases. The AWS CSP managed services tightly integrate with **AWS Security Hub** for cybersecurity monitoring of the factory in this reference design.

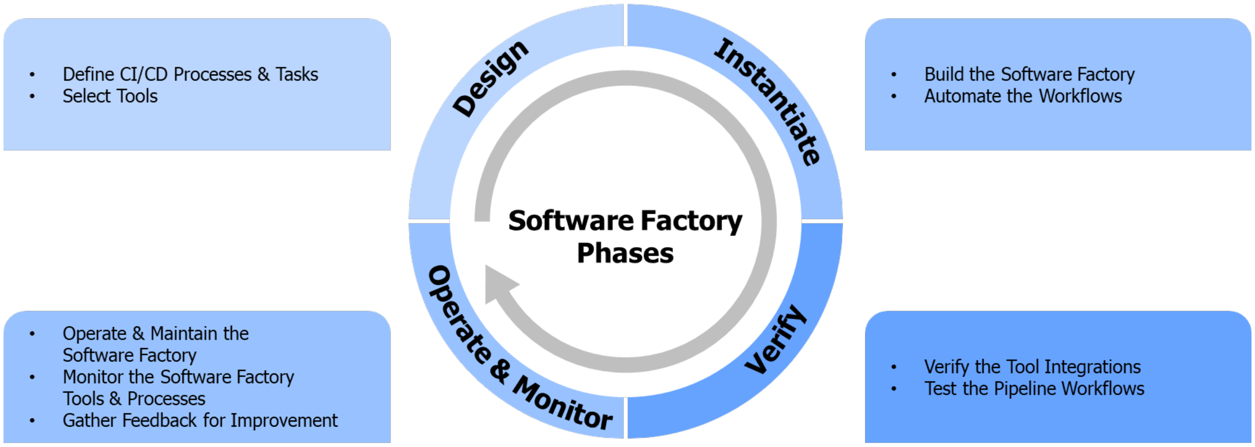

Define CI/CD Processes & Tasks
Select Tools
Design
Instantiate
Build the Software Factory
Automate the Workflows
Software Factory Phases
Operate & Maintain the Software Factory
Monitor the Software Factory Tools & Processes
Gather Feedback for Improvement
Operate & Monitor
Verify
Verify the Tool Integrations
Test the Pipeline Workflows

Figure 4: Software Factory Implementation Phases

The software factory leverages technologies and tools to automate the CI/CD pipeline processes defined in the DevSecOps lifecycle plan phase. There are no “one size fits all” or hard rules about what CI/CD processes should look like and what tools must be used. Each software team needs to embrace the DevSecOps culture and define processes that suit its software system architectural choices. The tool chain selection is specific to the software programming language choices, application type, tasks in each software lifecycle phase, and the system deployment platform.

The major components of this reference design’s software factory, EKS, ECR, and security related AWS services, are expected to be instantiated using the DoD Cloud IaC baselines from DISA. As of June, 2021, the following AWS IaC baselines have been created:

AWS DevOps Service:

CodeBuild
CodeCommit
CodePipeline

AWS Orchestration & Containers:

Elastic Kubernetes Service (EKS)
Fargate
Elastic Container Registry (ECR)

AWS Database Hosting:

Aurora
DynamoDB

AWS AI/ML:

SageMaker

AWS Serverless:

Lambda

AWS Supporting Services:

Audit Manager
CloudTrail
CloudWatch
Cognito
Config
Firewall
GuardDuty
Identity Access Manager
Key Management Service
Organizations
Security Hub
Service Catalog
Service Control Policies
Transit Gateway
Virtual Private Cloud (VPC)

DevSecOps teams create a pipeline workflow in the CI/CD orchestrator by specifying a set of stages, stage conditions, stage entrance and exit control rules, and stage activities. The CI/CD orchestrator automates the pipeline workflow by validating the stage control rules. If all the entrance rules of a stage are met, the orchestrator will transition the pipeline into that stage and perform the defined activities by coordinating the tools via plugins. If all the exit rules of the current stage are met, the pipeline exits out the current stage and starts to validate the entrance rules of the next stage. *Table 2* shows the features, benefits, and inputs and outputs of the CI/CD orchestrator.

Table 2: CD/CD Orchestrator Inputs/Outputs

Tool	Features	Benefits	Inputs	Outputs	Baseline	Managed Service
CI/CD orchestrator	Create pipeline workflow	Customizable pipeline solution	Human input about: • A set of stages • A set of event triggers • Each stage entrance and exit control gate • Activities in each stage	Pipeline workflow configuration	REQUIRED	**AWS CodePipeline**
	Orchestrate pipeline workflow execution by coordinating other plugin tools or scripts.	Automate the CI/CD tasks; Auditable trail of activities	Event triggers (such as code commit, test results, human input, etc.); Artifacts from the artifact repository	Pipeline workflow execution results (such as control gate validation, stage transition, activity execution, etc.); Event and activity audit logs		

4.1 Accessing the DoD Cloud IaC Baselines

The URL to access the baselines is:

https://code.il2.dso.mil/dod-cloud-iac

Detailed instructions for creating an account are contained in Appendix A at the end of this document.

4.2 Containerized Software Factory

Software factory tools include a CI/CD orchestrator, a set of development tools, and a group of tools that operate in different DevSecOps lifecycle phases. These tools are pluggable and must integrate into the CI/CD orchestrator. **In this reference design, instantiations must rely on a combination of DoD Cloud IaC baselines and a set of DevSecOps hardened containers accessed directly from Iron Bank.** Iron Bank containers are preconfigured and secured to reduce the certification and accreditation burden and are often available as a predetermined pattern or pipeline that will need limited or no configuration.

Running a CI/CD pipeline is a complex activity. Containerization of the entire CI/CD stack ensures there is no drift possible between different K8s cluster environments (development, test, staging, production). It further ensures there is no drift between different K8s cluster environments spanning multiple classification levels. Containerization also streamlines the update/accreditation process associated with the introduction and adoption of new DevSecOps tooling.

Figure 5 illustrates the containerized software factory reference design within the AWS CSP. The software factory is built on an underlying container orchestration layer powered by AWS EKS in an AWS cloud. For clarity, the software factory produces DoD applications and application artifacts as a product. Applications typically use different sets of hardened containers from Iron Bank than the ones used to create the software factory.

The software factory reference design illustrates how cybersecurity is weaved into the fabric of each factory pipeline. All of the tooling within the factory is based on hardened containers pulled from Iron Bank along with the pre-configured, pre-authorized DoD Cloud IaC baselines used to instantiate the various CSP managed services.

Moving from left to right, as code is checked into a branch triggering the CI/CD pipeline workflow and resulting automated build, SAST, DAST, and unit tests are executed, as the orchestrator coordinates different tools to perform various tasks defined by the pipeline. If the build is successful and a container image is defined, a container security scan is triggered. Some tests and security tasks may require human involvement or consent before being considered complete and passed. If all of these tests are successful, then the artifact is deployed into the test environment. If all of the entrance rules of the next stage are met, the orchestrator will transition the pipeline into that stage and perform the defined activities by coordinating the tools via plugins. When all stages are complete, a significant number of security activities have completed and the artifact is eligible for deployment into production. Deployment into production should be fully automated, but may be gated by a human actually pressing a button to trigger the deployment.

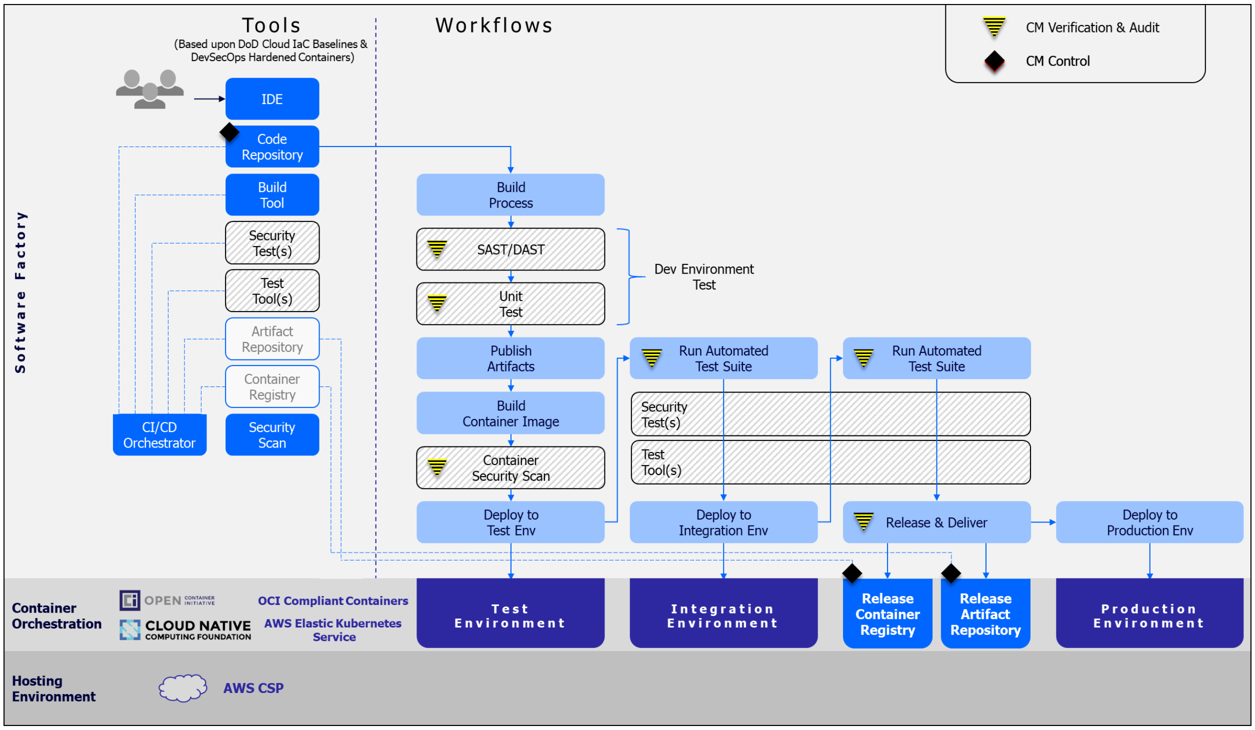


Figure 5: AWS CSP Software Factory Reference Design

DoD programs may have already implemented a DevSecOps platform. Operating a custom DevSecOps platform is an expensive endeavor because software factories require the same level of continuous investment as a software application. There are financial benefits for programs to plan a migration to a containerized software factory, reaping the benefits of centrally managed and hardened containers that have been fully vetted. In situations where a containerized software factory is impractical, or the factory requires extensive policy customizations, the program should consult with DoD CIO and (if applicable) its own DevSecOps program office to explore options and collaborate to create, sustain, and deliver program-specific hardened containers to Iron Bank.

5 Hosting Environment

The reference design was designed for Amazon Web Services as the CSP. The specific AWS environment, impact level 2, 4, 5, or 6, should be identified and selected by the mission program. The DoD Cloud IaC has been tested and piloted across each of these impact levels.

5.1 Container Orchestration

Software factory responsibilities include container orchestration, while the CSP is responsible for the underlying K8s control plane and resources (compute, storage, etc.). As described in the opening paragraphs of this section, this reference design mandates the use of AWS managed services provisioned using DoD Cloud IaC baselines, as illustrated in *Figure 6*.

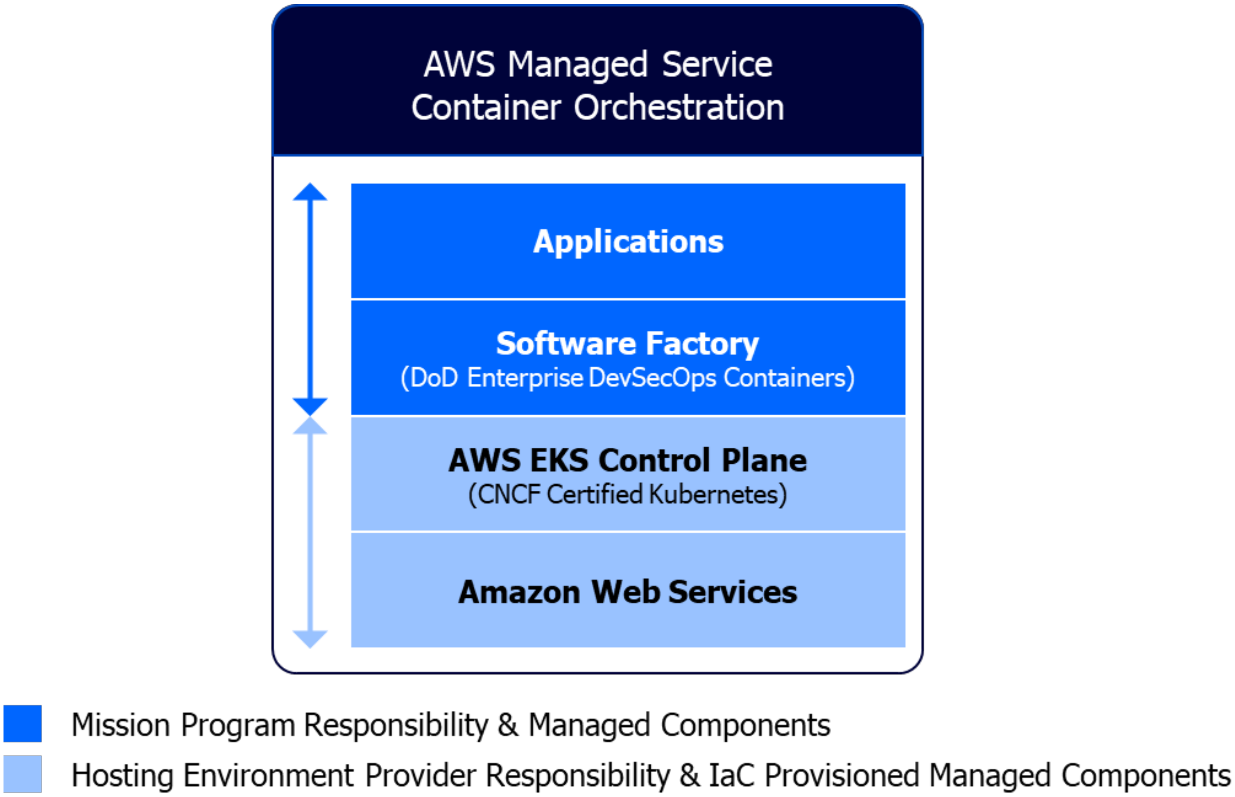

AWS Managed Service
Container Orchestration
Applications
Software Factory
(DoD Enterprise DevSecOps Containers)
AWS EKS Control Plane
(CNCF Certified Kubernetes)
Amazon Web Services
Mission Program Responsibility & Managed Components
Hosting Environment Provider Responsibility & IaC Provisioned Managed Components

Figure 6: DevSecOps Platform Options

It is the mission program's responsibility to avoid reconfiguration or modification of the IaC baselines that would create significant drift. The use of AWS managed services does not change the fact that the components are subject to monitoring and security control under the DoD policy in that hosting environment, such as the DoD Cloud Computing Security Requirements Guide (SRG) and DISA's Secure Cloud Computing Architecture (SCCA) for the Cloud environment.

A notional set of DevSecOps services, an abbreviated representation of the DevSecOps workflow, and various cybersecurity mechanisms are depicted in *Figure 7.*

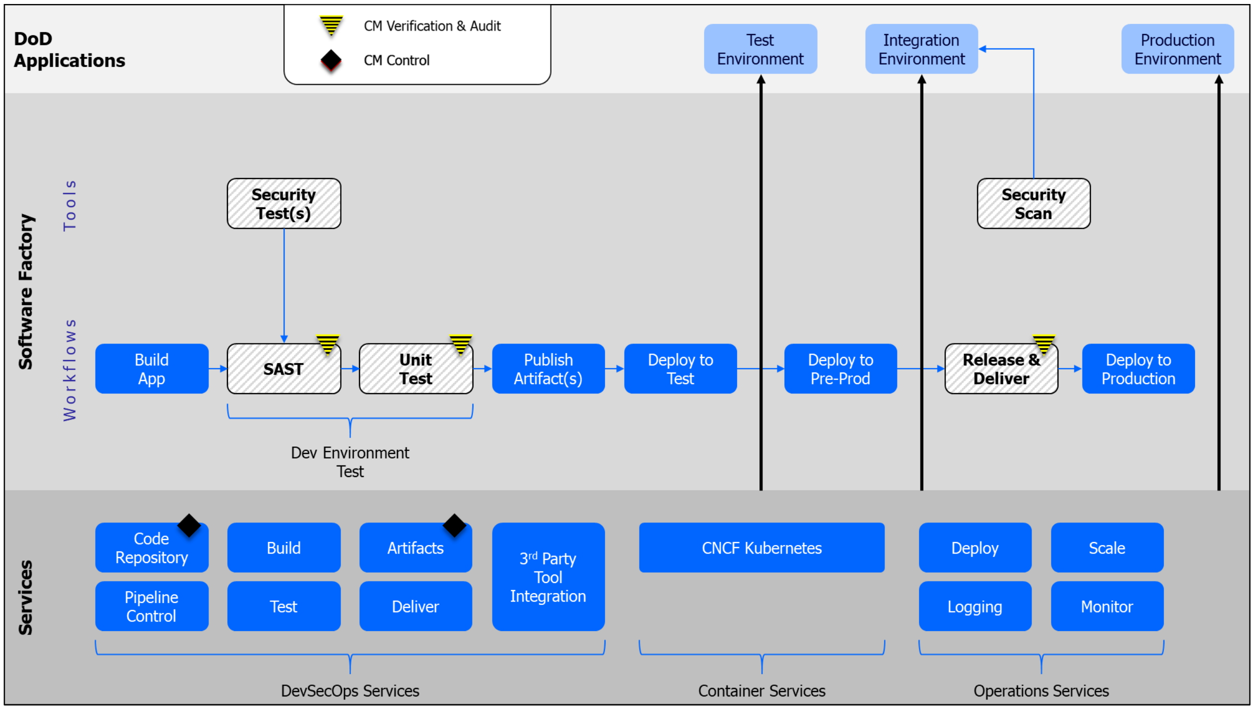


Figure 7: Software Factory - DevSecOps Services

6 Additional Tools and Activities

The **DevSecOps Tools and Activities Guidebook**, part of the DevSecOps Fundamentals, establishes common DevSecOps tools and activities. The guidebook recognizes that specific reference designs may elevate a specific tool from PREFERRED to REQUIRED, as well as add additional tools and/or activities that specifically support the nuances of a given reference design. The following sections identify those tools and activities unique to this reference design across the *Deploy* and *Monitor* phases of the DevSecOps lifecycle.

Table 3: Security Activities Summary and Cross-Reference

Activities	Phase	Activities Table Reference	Tool Dependencies	Tool Table Reference
Container or VM hardening	Develop	Table 4	Container security tool; Security compliance tool	
Container policy enforcement	Test	Table 6	Container policy enforcement	

Table 4: Develop Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Container image selection	Must leverage approved and hardened container images strictly from the Iron Bank repository	N/A	N/A	Artifact repo
Container hardening	Harden the deliverable for production deployment. Containers must follow the DISA Container Hardening Guide.[10]	Container	-Vulnerability report and recommended mitigation -Hardened Container & Build File	Container security tool

Table 5: Build Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline	Managed Service
Container builder	Build a container image based on a build instruction file. Must use a hardened container image from Iron Bank as the base image in all cases.	Container image build automation	Container base image; Container build file	OCI compliant container image	REQUIRED	**AWS CodeBuild**
Artifact Repository	Container Registry	Better quality software by using centrally managed, hardened containers.	Artifacts	Version controlled container	REQUIRED	**AWS ECR**

Table 6: Build Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Containerize	Packages all required OS components, developed code, runtime libraries, etc. into a hardened container	Container base image; Container build file	Container Image	Container Builder
Store artifacts	Store artifacts to the artifact repository	Container Image	Version controlled container image	Artifact Repository

Table 7: Test Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline	Managed Service
Container security tool	Container image scan OS check.	Ease the container hardening process	Container images or running containers	Vulnerability report and recommended mitigation.	REQUIRED	**AWS ECR**
Container policy enforcement	Support for Security Content Automation Protocol (SCAP) and Container configuration policies. These policies can be defined as needed.	Automated policy enforcement	Policies in SCAP form.	Compliance report	REQUIRED	**AWS Systems Manager** **AWS Lambda,** **AWS Event Bridge**
Example: https://aws.amazon.com/blogs/containers/compliance-as-code-for-amazon-ecs-using-open-policy-agent-amazon-eventbridge-and-aws-lambda/						
Security compliance tool	Scan and report for compliance regulations, such as DISA Security Technical Implementation Guides (STIGs), NIST 800-53.	Speed up ATO process.	Container images.	Vulnerability report and recommended mitigation.	PREFERRED	
Example: https://aws.amazon.com/blogs/security/how-to-automate-scap-testing-with-aws-systems-manager-and-security-hub/						

Table 8: Test Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependencies
Container policy enforcement	Check developed containers to be sure they meet container policies	Container, Policies in SCAP form	Container compliance report	Container policy enforcement

Table 9: Release and Deliver Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline	Managed Service
IaC / CaC	Automated "push button" instantiation of the applications running on K8s in addition to the software factory itself (including the SCSS stack on top)	Eliminate drift between environments; ensure desired state is always accurately captured in git.			REQUIRED	**AWS Code Commit** **AWS Code Pipeline** **AWS Cloud Formation**
GitOps Kubernetes Capability	Pull source code from git repositories instead of requiring the CI/CD pipeline to push artifacts to the next environment	Eliminates the need to open ports and/or require keys to be shared with CI/CD tooling. Eliminates environment drifts. Ensures desired state is always accurately captured in git.			RECOMMENDED	

Table 10: Release and Deliver Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Release go / no-go decision	This is part of configuration audit; Decision on whether to release artifacts to the artifact repository for the production environment.	Design documentation; Version controlled artifacts; Version controlled test reports; Security test and scan reports	go / no-go decision; Artifacts are tagged with release tag if go decision is made	CI/CD Orchestrator

Table 11: Deploy Phase Tools

Tool	Features	Benefits	Inputs	Outputs	Baseline	Managed Service
CNCF-certified K8s	Container grouping using pods; Health checks and self-healing Horizontal infrastructure scaling Container auto-scalability Domain Name Service (DNS) management Load balancing Rolling update or rollback; Resource monitoring and logging	Simplify operations by deployment and update automation Scale resources and applications in real time Cost savings by optimizing infrastructure resources	Container instance specification and monitoring policy	Running container	REQUIRED	**AWS EKS**
Service mesh	Ability to create a network of deployed services with load balancing, service-to-service authentication, and monitoring. Ability to enforce mTLS for east/west traffic.	Support for microservice interactions.	Control plane: service comms routing policies, authentication certificates. Data plane: service comms data	Control plane: service status reports Data plane: routed service communication data	REQUIRED	**AWS App Mesh**

Table 12: Deploy Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Deliver container to container registry	Upload the hardened container and associated artifacts to the container registry	Hardened container	New container instance	**AWS EKS** **AWS ECR**

Table 13: Operate Phase Activities

Activities	Description	Inputs	Outputs	Tool Dependency
Scale	Scale manages containers as a group. The number of containers in the group can be dynamically changed based on the demand and policy.	Real-time demand and container performance measures Scale policy (demand or Key Performance Indicator (KPI)threshold; minimum, desired, and maximum number of containers)	Optimized resource allocation	**AWS EKS**
Load balancing	Load balancing equalizes the resource utilization	Load balance policy Real time traffic load and container performance measures	Balanced resource utilization	**AWS EKS**

Table 14: Monitor Phase Tools

Tool	Features	Benefits	Baseline
Resource, Service, Container policy enforcement	Support for Security Content Automation Protocol (SCAP) and container configuration policies. These policies can be defined as needed.	Automated policy enforcement	REQUIRED
Vulnerability Management	Provides vulnerability management	Makes sure everything is properly patched to avoid known vulnerabilities	REQUIRED
CVE Service / Host Based Security	Provides CVEs. Used by the vulnerability management agent in the security sidecar container.	Makes sure the system is aware of known vulnerabilities in components.	REQUIRED

Table 15: AWS CSP Managed Service Monitoring Tools

Tool	Features	Benefits	Baseline
Netflow Analysis	Logs network traffic within as enclave Network troubleshooting	Helps to find anomalous patterns across environment and Platform	REQUIRED
Centralized Logging	Stores logs from the entire environment. Used by the SEIM/SOAR for log analysis and incident detection	Place to store logs across environment and Platform	REQUIRED
Centralized Analysis	SEIM/SOAR for log analysis and incident detection Tier 3 CSSP tools	Helps to find anomalous patterns across environment and Platform	RECOMMENDED

6.1 Continuous Monitoring in K8s

Continuous monitoring of a K8s cluster must include behavior and signature-based detection in the runtime environment. These and other container specific controls are captured in NIST Special Publication 800-190, *Application Container Security Guide*. CSP services also routinely monitor and scan CSP resources and services for misconfiguration, incorrect access control, and security events. These CSP specific capabilities should be integrated into every continuous monitoring strategy.

Figure 8 illustrates a notional process of monitoring, logging, and log analysis and alerting. The process starts with application logging, compute resource monitoring, storage monitoring, network monitoring, security monitoring, and data monitoring at the Kubernetes pod level (the individual subsystem level in the case of VM deployment and the service level for serverless deployments).

Each application will need to determine how it is divided into subsystems, the number of subsystems, and the specific monitoring mechanisms within the subsystems. The security tools within each subsystem will aggregate and forward the event logs gathered from monitoring to a locally centralized aggregated logs database on the mission program platform. The aggregated logs will be further forwarded to the Logs/Telemetry Analysis in the Defensive Cyber Operations / Tier 2 CSSP after passing the program application configured log filter. The program's local log SIEM/SOAR analysis capability will analyze the aggregated logs and generate incident alerts and reports.

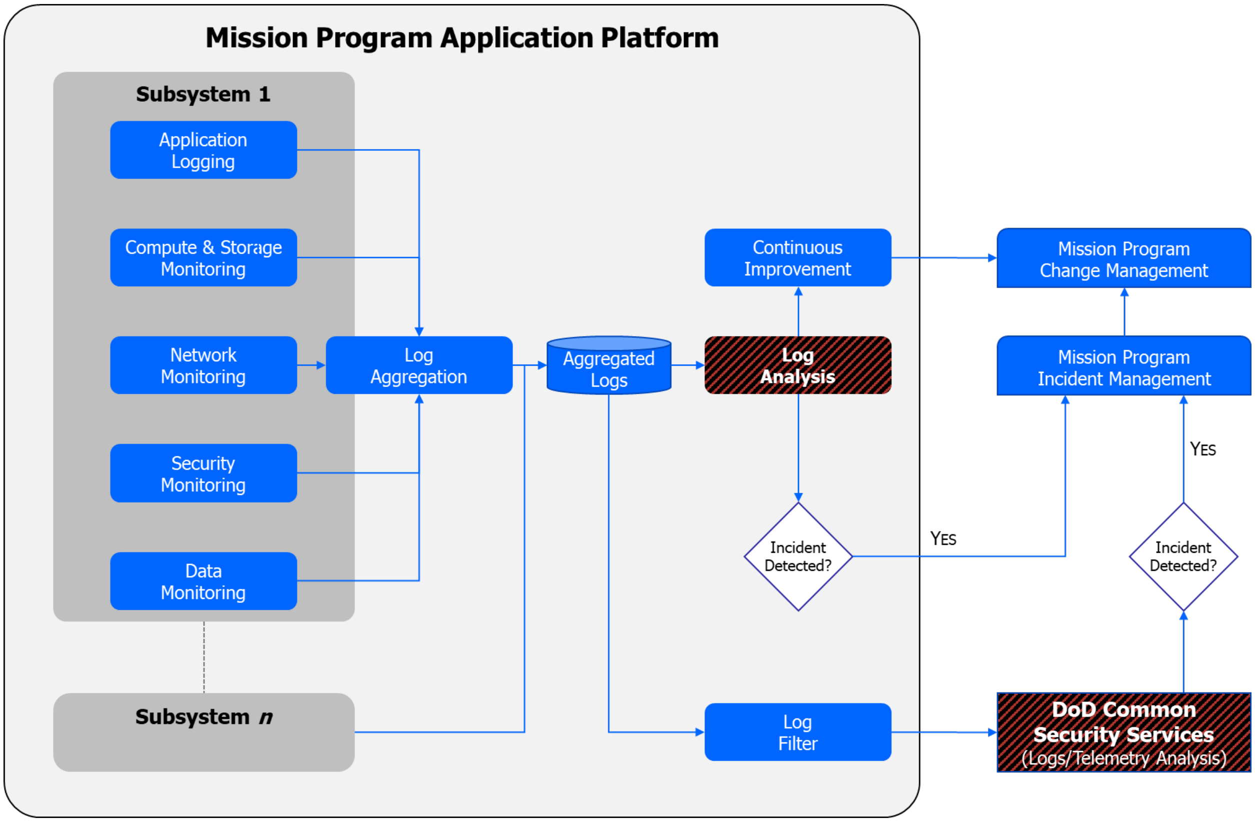


Figure 8: Logging and Log Analysis Process

Incidents will be forwarded to the mission program incident management system to facilitate change request generation for incident resolution. The mission program incident management should alert or notify the responsible personnel about the incidents. The change request may be created to address the incident. These actions make the DevSecOps pipeline a full closed loop from secure operations to planning.

6.1.1 CSP Managed Services for Continuous Monitoring

The use of CSP managed services for monitoring alongside 3rd party security tools should always be viewed through a "both/and" lens instead of an "either/or" lens. CSP managed services can be utilized to monitor CSP resources & services, netflow, and entity behavior analysis at a deeper level than with 3rd party tools alone. It may also be possible to employ CSP managed services to perform log analysis (SEIM/SOAR). The monitoring ecosystem should rely on curated IaC to instantiate the monitored environment to the maximum extent possible, ensuring completeness and accelerating the A&A process.

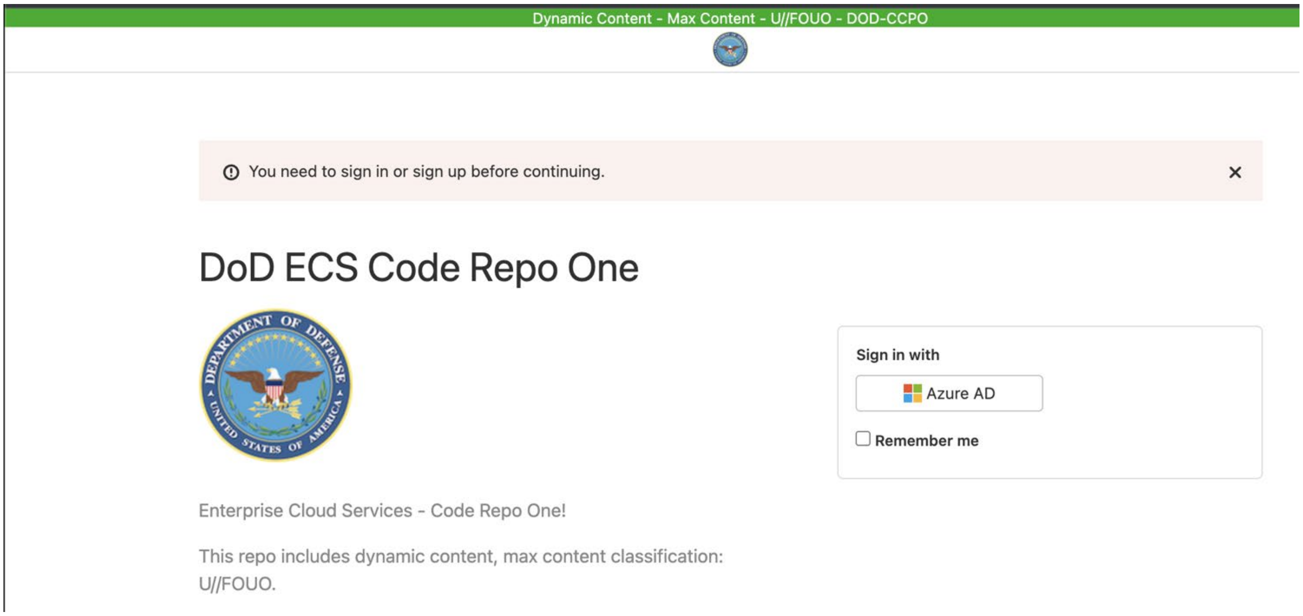
7 Appendix A: Accessing the DoD Cloud IaC Code Repository

Prerequisite:

- You must have an entry in Global Directory to access the repository. This can be tested by attempting to access DoD Teams (https://dod.teams.microsoft.us/). If you are unable to log into this site, you likely will not be able to access the repository.

Steps:

1. Navigate to https://code.apps.core.ecs.mil/USG/DOD/DISA/CCPO/dod-cloud-iac/
2. You will see a login screen as shown below. With your CAC inserted, select "Azure AD" to login through Global Directory



Dynamic Content - Max Content - U//FOUO - DOD-CCPO

You need to sign in or sign up before continuing.

DoD ECS Code Repo One

Sign in with

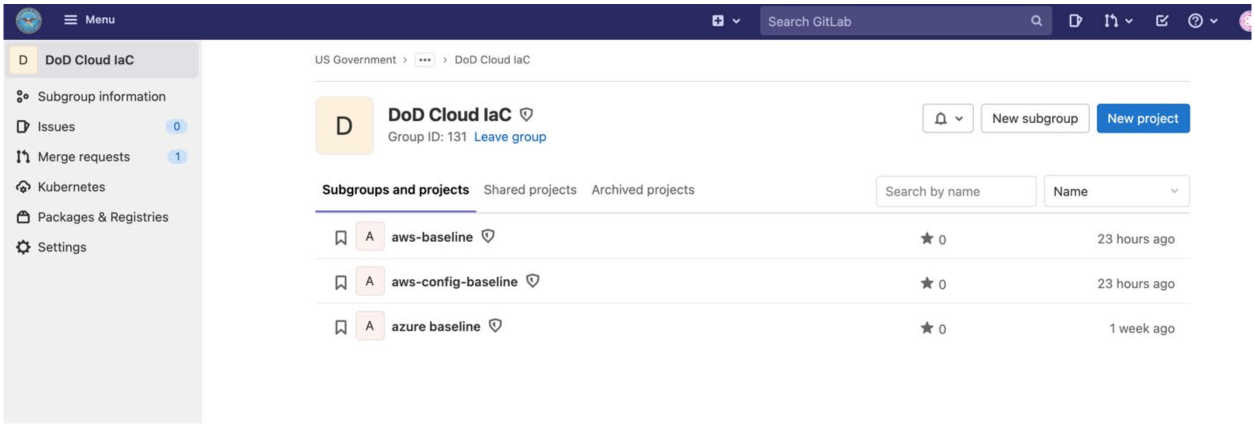
Azure AD

□ Remember me

Enterprise Cloud Services - Code Repo One!

This repo includes dynamic content, max content classification: U//FOUO.

3. Follow the prompts and agree to the terms of service
4. Once logged in, you should be redirected to the page shown below



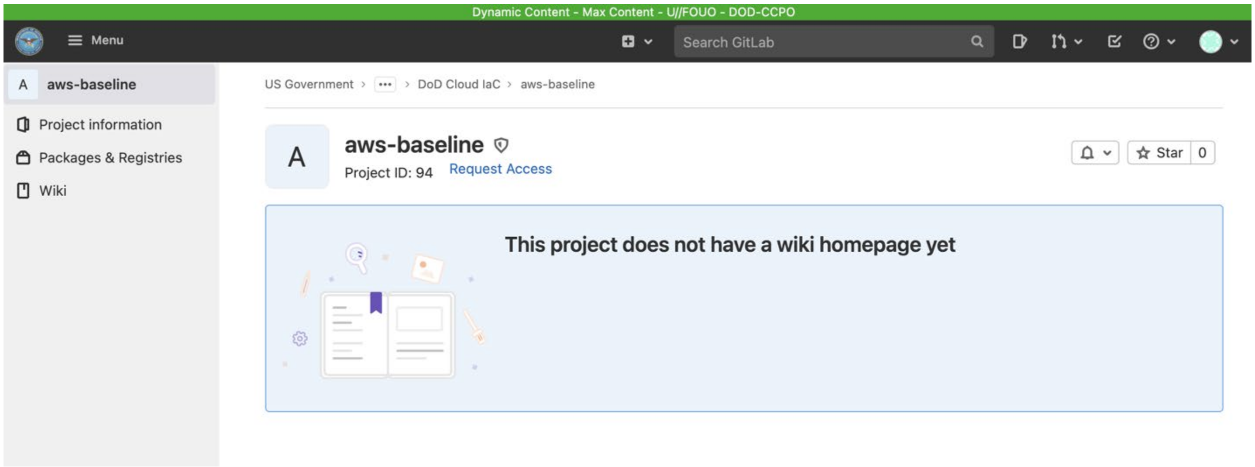
US Government > ··· > DoD Cloud IaC

DoD Cloud IaC

Group ID: 131 Leave group

A aws-baseline	★ 0	23 hours ago
A aws-config-baseline	★ 0	23 hours ago
A azure baseline	★ 0	1 week ago

5. Click on the project you would like to access. In this example aws-baseline was selected.

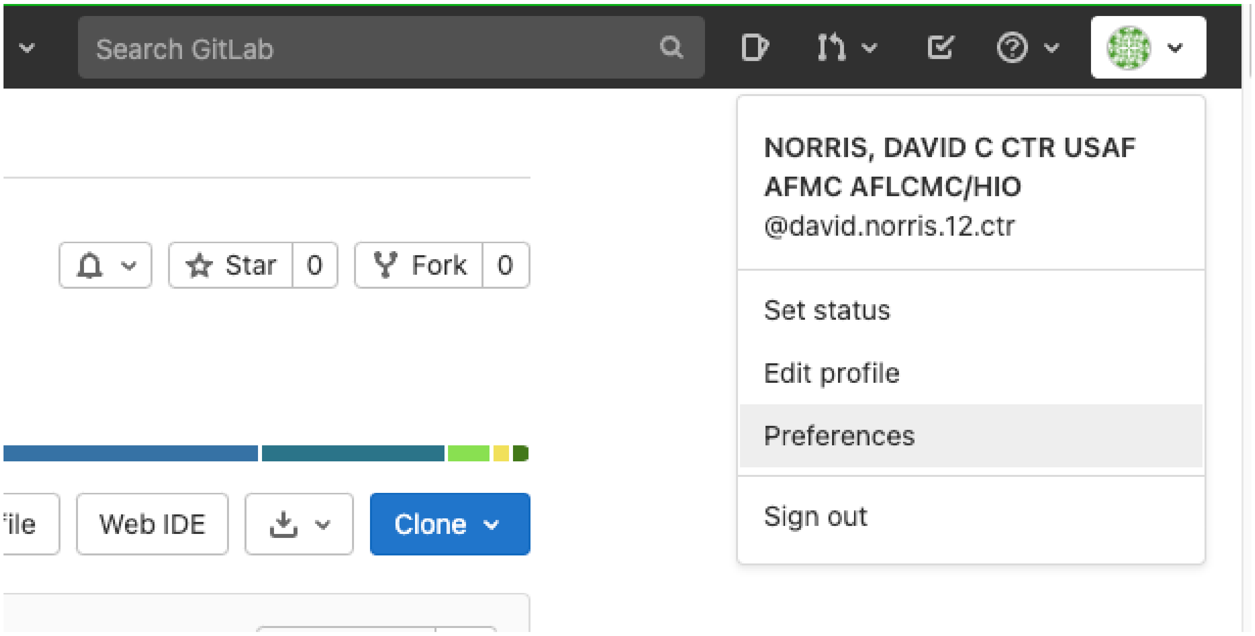


This project does not have a wiki homepage yet

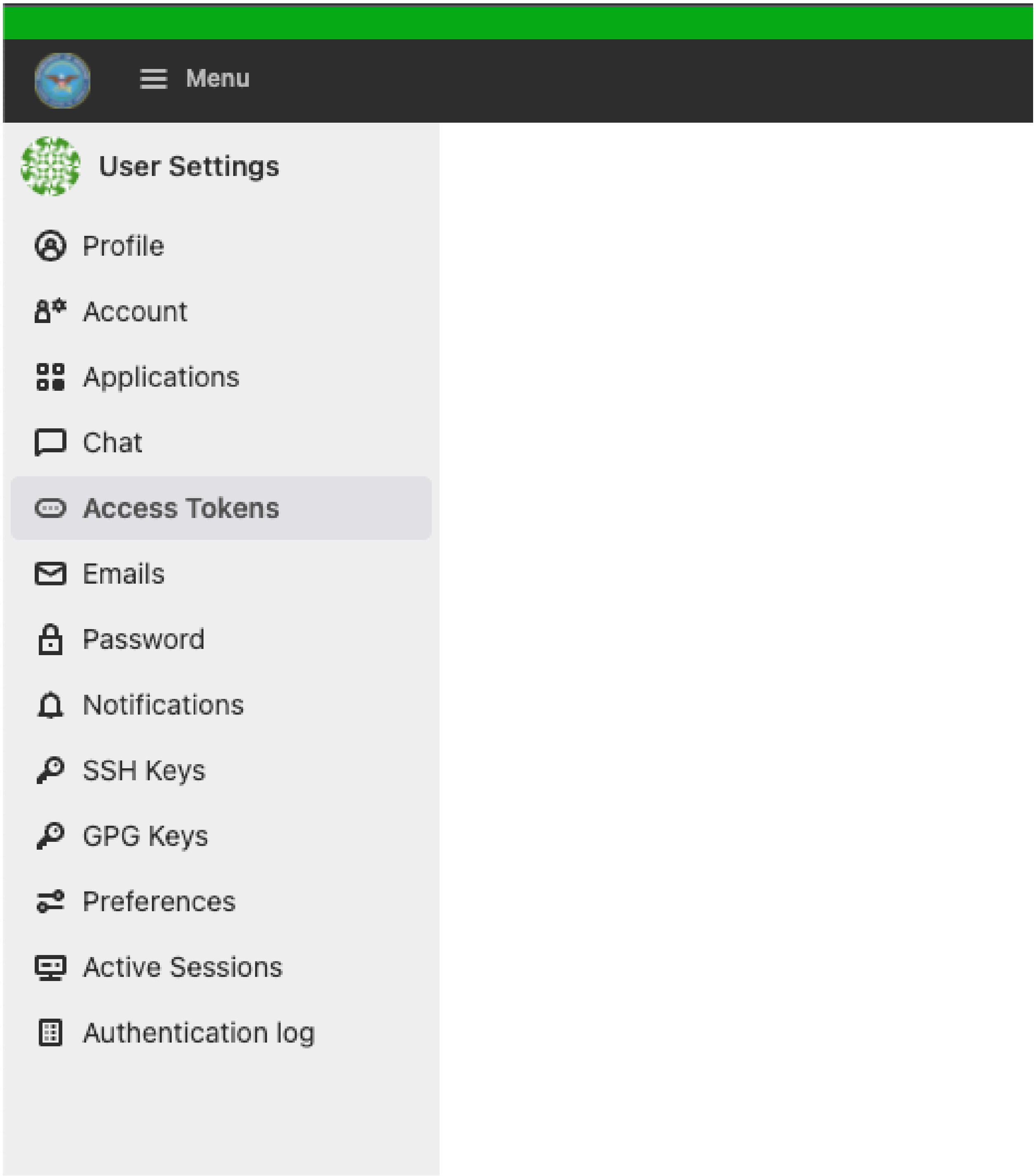
6. Select “Request Access”
7. An email will be sent to the repository maintainers alerting them to your request
8. Access will be granted once the team confirms your request

<u>Creating an Access Token to Pull the Repository (Approved Developer Only)</u>

1. Select the user profile icon in the top right corner of the screen and choose “Preferences”



2. Select “Access Tokens” from the menu on the left side of the page



3. Enter a name for the token, the expiration date and the required scopes (for most, this will only include "read_repository")
4. Hit "Create personal access token" and save the new token
5. When cloning the repository, this token will serve as the password when prompted for login on the git client