KUBERNETES
SECURITY TECHNICAL IMPLEMENTATION GUIDE
(STIG) OVERVIEW

Version 1, Release 5

27 April 2022

Developed by DISA for the DoD

Trademark Information

TABLE OF CONTENTS

Page

LIST OF TABLES

Page

LIST OF FIGURES

Page

1. INTRODUCTION

1.1 Executive Summary

The Kubernetes Security Technical Implementation Guide (STIG) provides technical requirements for securing a basic Kubernetes platform version 1.16.7 and newer. A basic Kubernetes cluster is composed of a Kubernetes master, application programming interface (API) server, scheduler, controllers, etcd, and the worker nodes. There can be other components within a Kubernetes installation; however, those listed here are the basic components within every installation and are the ones covered in this STIG.

Other components, such as a runtime and a container network interface (CNI), act differently depending on the installed software (runtime examples are Docker, containerd, rkt, and lxd) or plugin (CNI plugin examples are Flannel, Calico, Canal, and Weave Net). The component also determines what additional security can be implemented for Kubernetes. For instance, the CNI installed can determine if network policies can be implemented and what type of policies they are. Because of the differences in capacities, features of Kubernetes components are outside the scope of the Kubernetes STIG, but the components do need to be secured. To secure the components outside the scope of this document, use the specific vendor STIG or technology Security Requirements Guide (SRG).

Kubernetes is also a portable, extensible, open-source platform for managing containerized workloads and services that facilitates both declarative configuration and automation. With the platform, services such as a DNS, firewall, router, and web console may also be deployed. These services are also outside the scope of this document and must follow the appropriate vendor-specific STIG, if one exists. If a vendor-specific STIG does not exist, the more generic technology SRG must be used. Services must also follow any guidance that pertains to the way the services are implemented.

1.2 Authority

Department of Defense Instruction (DoDI) 8500.01 requires that "all IT [information technology] that receives, processes, stores, displays, or transmits DoD information will be […] configured […] consistent with applicable DoD cybersecurity policies, standards, and architectures." The instruction tasks that DISA "develops and maintains control correlation identifiers (CCIs), security requirements guides (SRGs), security technical implementation guides (STIGs), and mobile code risk categories and usage guides that implement and are consistent with DoD cybersecurity policies, standards, architectures, security controls, and validation procedures, with the support of the NSA/CSS [National Security Agency/Central Security Service], using input from stakeholders, and using automation whenever possible." This document is provided under the authority of DoDI 8500.01.

Although the use of the principles and guidelines in these SRGs/STIGs provides an environment that contributes to the security requirements of DoD systems, applicable NIST SP 800-53 cybersecurity controls must be applied to all systems and architectures based on the Committee on National Security Systems (CNSS) Instruction (CNSSI) 1253.

1.3 Vulnerability Severity Category Code Definitions

Severity Category Codes (referred to as CAT) are a measure of vulnerabilities used to assess a facility or system security posture. Each security policy specified in this document is assigned a Severity Category Code of CAT I, II, or III.

Table 1-1: Vulnerability Severity Category Code Definitions

	DISA Category Code Guidelines
CAT I	Any vulnerability, the exploitation of which will **directly and immediately** result in loss of Confidentiality, Availability, or Integrity.
CAT II	Any vulnerability, the exploitation of which **has a potential** to result in loss of Confidentiality, Availability, or Integrity.
CAT III	Any vulnerability, the existence of which **degrades measures** to protect against loss of Confidentiality, Availability, or Integrity.

1.4 STIG Distribution

Parties within the DoD and federal government's computing environments can obtain the applicable STIG from the DoD Cyber Exchange website at https://cyber.mil/. This site contains the latest copies of STIGs, SRGs, and other related security information. Those without a Common Access Card (CAC) that has DoD Certificates can obtain the STIG from https://public.cyber.mil/.

1.5 Document Revisions

All technical NIST SP 800-53 requirements were considered while developing this STIG. Requirements that are applicable and configurable will be included in the final STIG. A report marked Controlled Unclassified Information (CUI) will be available for items that did not meet requirements. This report will be available to component authorizing official (AO) personnel for risk assessment purposes by request via email to: disa.stig_spt@mail.mil.

1.6 Other Considerations

DISA accepts no liability for the consequences of applying specific configuration settings made on the basis of the SRGs/STIGs. It must be noted that the configuration settings specified should be evaluated in a local, representative test environment before implementation in a production environment, especially within large user populations. The extensive variety of environments makes it impossible to test these configuration settings for all potential software configurations.

For some production environments, failure to test before implementation may lead to a loss of required functionality. Evaluating the risks and benefits to a system's particular circumstances and requirements is the system owner's responsibility. The evaluated risks resulting from not

applying specified configuration settings must be approved by the responsible AO. Furthermore, DISA implies no warranty that the application of all specified configurations will make a system 100 percent secure.

Security guidance is provided for the DoD. While other agencies and organizations are free to use it, care must be given to ensure that all applicable security guidance is applied at both the device hardening level and the architectural level due to the fact that some settings may not be configurable in environments outside the DoD architecture.

1.7 Product Approval Disclaimer

The existence of a STIG does not equate to DoD approval for the procurement or use of a product.

STIGs provide configurable operational security guidance for products being used by the DoD. STIGs, along with vendor confidential documentation, also provide a basis for assessing compliance with cybersecurity controls/control enhancements, which supports system assessment and authorization (A&A) under the DoD Risk Management Framework (RMF). Department of Defense AOs may request available vendor confidential documentation for a product that has a STIG for product evaluation and RMF purposes from disa.stig_spt@mail.mil. This documentation is not published for general access to protect the vendor's proprietary information.

AOs have the purview to determine product use/approval in accordance with (IAW) DoD policy and through RMF risk acceptance. Inputs into acquisition or pre-acquisition product selection include such processes as:

- National Information Assurance Partnership (NIAP) evaluation for National Security Systems (NSS) (https://www.niap-ccevs.org/) IAW CNSSP #11
- National Institute of Standards and Technology (NIST) Cryptographic Module Validation Program (CMVP) (https://csrc.nist.gov/groups/STM/cmvp/) IAW Federal/DoD mandated standards
- DoD Unified Capabilities (UC) Approved Products List (APL) (https://www.disa.mil/network-services/ucco) IAW DoDI 8100.04

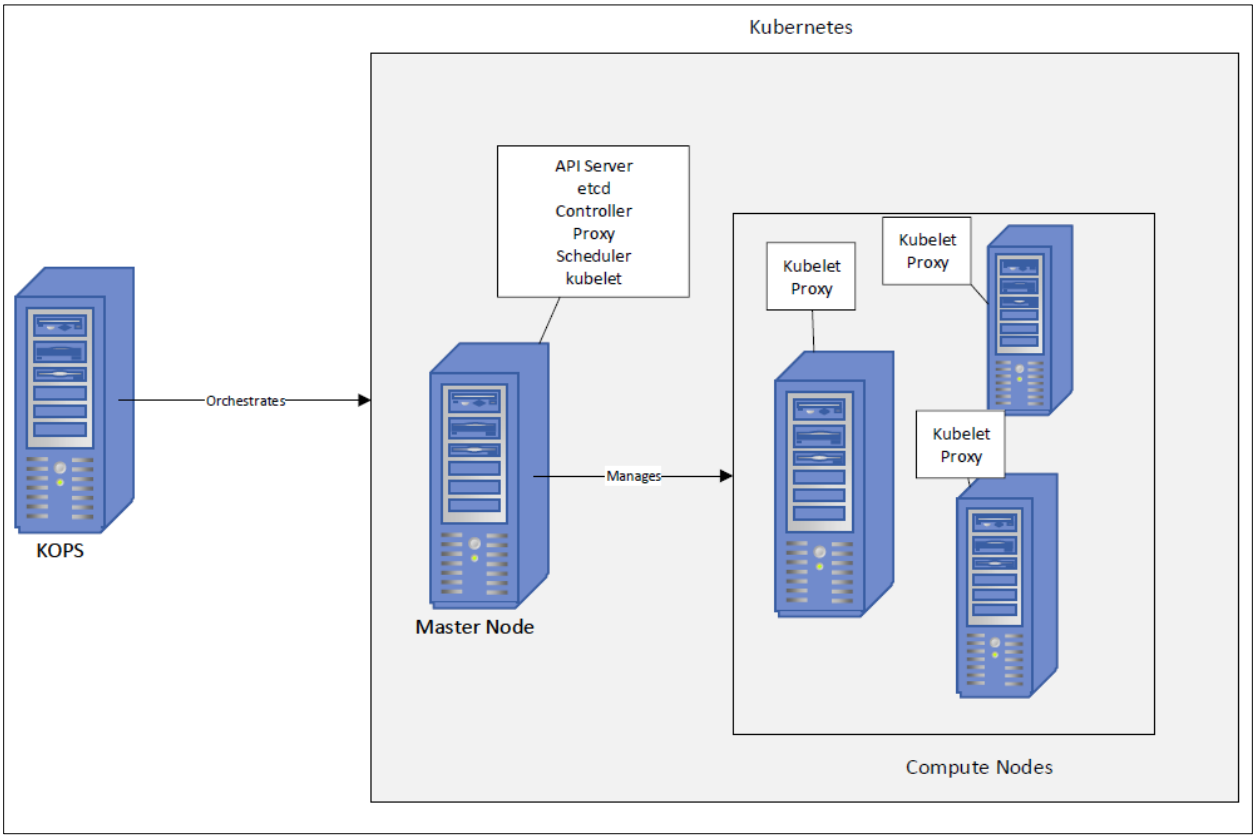
2. ASSESSMENT CONSIDERATIONS

2.1 Security Assessment Information

Kubernetes is a platform that can be installed in many different configurations, but the services that make up Kubernetes remain the same. For development of the Kubernetes STIG, the environment was orchestrated using Kubernetes Operations (KOPS). This means that many of the components, such as the API server, etcd, controller, proxy, and scheduler, are pods. Kubelet is an agent that runs on each node and ensures that containers are running. Kube-proxy is a network proxy that runs on each node in the cluster to maintain network rules and allow network communication to pods. Kube-Proxy uses the Linux system packet filtering layer.

Because Kubernetes can be deployed in many different configurations, some configurations load these components on dedicated nodes for performance or security reasons. This indicates that many of the checks and fixes within the STIG may return no result or the file and location are non-existent. It is important to remember that this document is an implementation guide. The requirements and the settings to implement and secure the Kubernetes cluster are the same regardless of the implementation; the difference is where the setting might be located.

Figure 2-1: KOPS Orchestration of Kubernetes

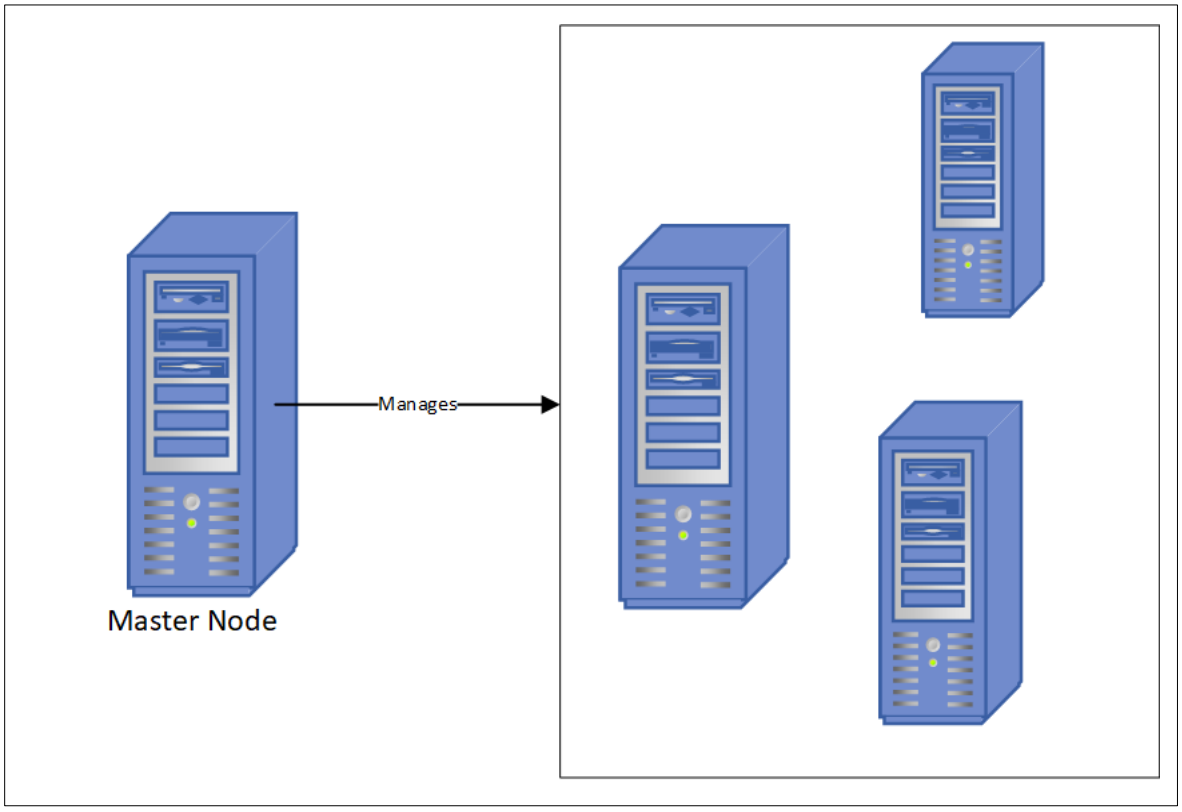


3. CONCEPTS AND TERMINOLOGY CONVENTIONS

Kubernetes is an open-source container orchestration platform. The goal is to provide a platform for easy orchestration, management, and scaling of containers that offer services for the end user. Kubernetes is used in many products as the underlying technology. Vendor products offering overlays to Kubernetes provide advanced services and utilities such as an improved system administrator, developer, and end user experience.

To understand Kubernetes, it is important to look at the terminology used within Kubernetes and what these terms reference. To begin with, the basic architecture should be understood. A basic Kubernetes installation consists of a master node and multiple worker nodes. This structure is often called the Kubernetes Cluster. The master node is responsible for exposing the API, scheduling deployments and managing the overall cluster. The worker node is responsible for having a runtime to execute containers and an agent to communicate with the master. Other services may be available on the worker node to perform tasks such as logging, monitoring, and service discovery. The worker nodes offer computing, networking, and storage resources to the running containers. These nodes are the workhorse of the Kubernetes cluster.

Figure 3-1: Basic Kubernetes Architecture



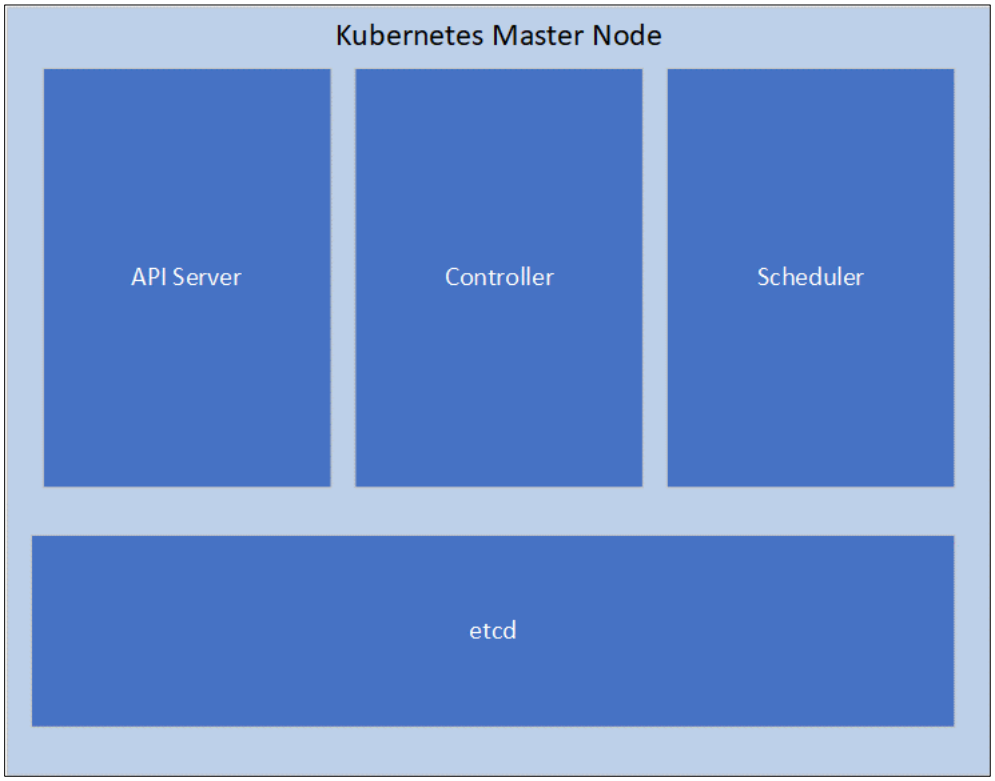
As mentioned previously, Kubernetes is an orchestrator of containers. Container images, containers, and pods are all interrelated. The container image, often referred to as “the image”, is an unchangeable, static file that contains executable code. When this file is instantiated or executed within Kubernetes, it becomes a container. The term “pod” is used for a collection of

one or more containers. The pod acts as a logical boundary for containers sharing the same context and resources.

3.1 Master Node Services

To manage the Kubernetes cluster, the master node will have several services. Each service is responsible for a single task. These services are etcd, API server, controller, and scheduler.

Figure 3-2: Kubernetes Master Node Services



3.1.1 API Server

The API server services Representational State Transfer (REST) operations and provides the front end to the shared state of the cluster. All components within the Kubernetes cluster communicate through the API server. Some examples of API objects that can interact through the API server are pods, services, and replication controllers.

3.1.2 etcd

The Master node uses etcd as a distributed key-value database, which acts as the single source of truth for all components of the Kubernetes cluster. The master queries etcd to retrieve the current state of nodes, pods, and containers.

3.1.3 Controller

The Master node uses the controller to watch the shared state of the cluster through the API server. The controller is constantly attempting to move the current state of the cluster toward the desired state.

3.1.4 Scheduler

The scheduler uses policy, topology information, and workload data to make availability, performance, and capacity decisions. When the scheduler discovers a new pod that has not been assigned to a node, the scheduler becomes responsible for determining the best node on which the pod will run.

3.2 Node Services

On every node, master, and worker, there are services used for basic network capabilities and communication between the nodes. These services are the proxy and kubelet.

3.2.1 Proxy

The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API for each node. These services can perform simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of back ends. The user must create a service with the apiserver API to configure the proxy.

3.2.2 kubelet

The kubelet service is the primary node agent that runs on each node. The kubelet registers the node with the API server. The primary communication to the kubelet is through the API server using a pod specification (PodSpec) to describe each pod. The kubelet ensures that the pods described in each PodSpec meet the described specifications, are running, and are healthy.

4. GENERAL SECURITY REQUIREMENTS

Kubernetes security goes beyond configuration settings. To secure Kubernetes properly, consideration must be given to the services being hosted, who the user community is, what type of data is being accessed, and where Kubernetes will reside. By not looking beyond Kubernetes itself, security flaws in the implementation can lead to the compromise of user personally identifiable information (PII), organization-sensitive data and processes, and access to other systems and applications within the organization with a trusted relationship to Kubernetes services.

4.1 Hosting Operating Systems

The operating system is the foundation for the Kubernetes cluster. By not securing the operating system properly, the cluster can become a front end for a nefarious user to gain access to an organization's networked resources.

When securing the Kubernetes cluster, care should be taken to secure files and set user roles and privileges to the least required for proper operating system and Kubernetes operation. The Kubernetes STIG addresses operating system file permissions for Kubernetes files, but within an operating system, there are settings and processes outside the realm of the Kubernetes STIG. There may also be instances where a Kubernetes requirement can be met, but without the operating system requirement being met, the cluster is not fully secure.

While most Kubernetes STIG settings will occur on the master node, it is important to remember that the cluster includes worker nodes, and these nodes also need to have their operating systems secured. To secure the operating systems on the master and worker nodes properly, use the appropriate operating system SRG or specific vendor STIG.

4.1.1 Roles

Defining user roles properly is essential to securing the Kubernetes cluster. Too often, all the operating system users are given the same roles. Giving users more privileges than necessary allows a user to escalate their privileges and make Kubernetes cluster changes, which is an administrator function. It is crucial to look at the roles the organization wants to implement for privileged users and give users only the roles required for carrying out their duties. The definition and duties of each role should be completed before any user accounts are created and the Kubernetes cluster is deployed.

4.1.2 File Permissions

When securing an operating system, many of the requirements rely on privileges and ownership of files. The permissions laid forth by the operating system requirements may or may not be stricter than those of the Kubernetes STIG requirements for privileges and ownership. Care must be taken to give the least privileges and ownership to Kubernetes files and still allow operation of Kubernetes and the hosted services. To arrive at the proper least privilege settings for

Kubernetes and hosted services, a test environment should be used along with a well-developed test plan to ensure the production Kubernetes operates properly and as expected.

4.2 Kubernetes Management

Kubernetes cluster management is the process of providing administrative duties in the configuration, deployment, and sustainment of the Kubernetes software, components, user services, and nodes. The management duties can be performed through local (i.e., console) or remote access. Remote access can take many forms, such as through the internet or a dedicated management network.

When the management is done locally, the hosting hardware and operating system perform the validation of users, assignment of permissions or privileges to the user, and enforcement of file protections. Constraining the user to only files and functions needed to perform their duties is the major Kubernetes cluster security concern.

Remote access has the added security concern of data transmission. All remote management to Kubernetes must be encrypted. The encryption of the traffic should begin at the start of the transmission session. The loss of administrative credentials during a non-encrypted session would negate any security that encryption of later traffic would add. Some methods of performing administrative activities remotely are through third-party software used specifically to administer Kubernetes (e.g., web console), secure shells and virtual private networks (VPNs), or a dedicated management network.

Remote access must also be controlled so it is not easily available or viewable by non-administrative users. Where local access can be controlled through physical barriers, remote access needs to be controlled through electronic barriers such as access lists or management networks. Care should be taken not to bypass security measures already in place to protect the Kubernetes cluster when implementing remote access technologies.

No matter the method used for Kubernetes cluster management, users must be authenticated using common access card (CAC) credentials. The validation of CAC credentials will not be done by Kubernetes but by the operating system or local and remote access host system access controls. If a web console is provided, the web service must authenticate the users via CAC authentication before granting access to the web console.

4.3 Kubernetes Component Authentication

Trusted and secure communication between Kubernetes components is essential. In addition to having a secure connection, the relationship must be trusted. For a component to identify itself, certificates are used. These certificates must be protected and tied to a trusted certificate authority (CA). By using only certificates from a trusted CA, the use of self-signed certificates is not permitted. Within a DoD environment, the DoD should be the CA and issue the certificates.

4.4 Transmitted Data Protection

Transmitted data must be protected, whether between the user and a service within Kubernetes, intercomponent communication, or data image pulls. If an adversary were able to compromise the data, the entire Kubernetes cluster would be compromised. A common method of securing communications is the use of a protocol that provides data integrity and encryption services. Transport Layer Security (TLS) is more effective than Secure Sockets Layer (SSL) encryption, improving privacy and data security for communications between applications. Kubernetes must use TLS v1.2 at a minimum to secure the Kubernetes cluster.

4.5 Conclusion

Kubernetes is an ecosystem within itself, made up by the master and worker nodes; Kubernetes components such as DNS servers, firewalls and routers; and user services. Securing the overall Kubernetes cluster must take into consideration each of these parts, the communication between the parts, the user community, and the data being processed. This STIG does not address every component or service offered or needed by the overall Kubernetes cluster because existing security documentation within SRGs, STIGs, and guides address them. Following are some questions that can be asked to help further secure Kubernetes:

- Are the user container images following application guides and business best practices?
- Are the user container images using industry-standard formats such as those outlined at the Open Container Initiative (OCI) to allow for better security tool scanning?
- Are all the components and services hardened according to guidance within the technology SRG or more specific technology STIG? If a STIG or SRG is not available, are industry standards being used such as those at the Cloud Native Computing Foundation (CNCF)?
- Are all components and services following ports and protocol guidelines set forth by DoD Instruction 8551.01?
- Are all endpoints trusted and communication channels encrypted?
- Are workloads, user groups, and data sensitivity levels being isolated properly?
- Is there a process for introducing new container images into the production environment?
- Are the security tools monitoring the components and user services container aware and designed to operate at the scale and change rate typically seen with containers?
- Are the containers running with policies to limit resource usage such as CPU, storage, and memory?

To fully secure the Kubernetes cluster, the system administrators must fully understand how the Kubernetes cluster is installed and the services and data Kubernetes will host. This goes beyond the Kubernetes STIG but is essential to guarantee an overall secure system.