Harrisburg University of Science and Technology
Digital Commons at Harrisburg University

Dissertations and Theses

Computer and Information Sciences, Graduate (CSMS)

Fall 12-14-2022

Cloud Container Security' Next Move

Vishakha Sadhwani
vsadhwani@my.harrisburgu.edu

Follow this and additional works at: https://digitalcommons.harrisburgu.edu/csms_dandt

 Part of the Computer Sciences Commons, and the Risk Analysis Commons

Recommended Citation

Sadhwani, V. (2022). *Cloud Container Security' Next Move*. Retrieved from https://digitalcommons.harrisburgu.edu/csms_dandt/3

This Thesis is brought to you for free and open access by the Computer and Information Sciences, Graduate (CSMS) at Digital Commons at Harrisburg University. It has been accepted for inclusion in Dissertations and Theses by an authorized administrator of Digital Commons at Harrisburg University. For more information, please contact library@harrisburgu.edu.

Cloud Container Security' Next Move:

Defining Architecture and Implementing Automation for Container Workflows with a Secure Infrastructure on Cloud.

by

Vishakha Sadhwani

Thesis submitted to the Faculty of the Graduate School of the

in fulfillment of the requirements for the degree of

Master of Science in Computer and Information Sciences

Supervised by: Mike Shahine, Ph.D.

[2023]

Table of Contents

Abstract

In the last few years, it is apparent to cybersecurity experts everywhere that the proverbial container tech genie is out of the bottle, and has been widely embraced across multiple organizations. To achieve the flexibility of building and deploying applications anywhere and everywhere, cloud native environments have gained great momentum and made the development lifecycle simpler than ever. However, container environments brings with them a range of cybersecurity issues that includes images, containers, hosts, runtimes, registries, and orchestration platforms, which needs the necessity to focus on investing in securing your container stack.

According to this report[1], released by cloud-native security provider Aqua Security on June 21', there are multiple ways attackers can breach a company's container infrastructure and the image supply chain. They also estimated a rise of 600% in the second half of 2020 as compared to the previous year. These attacks generally involves passive scanning methods to access servers that run Docker and Kubernetes platform. Per another report[2], the focus should not only be around securing the tools that are cloud provided, but should also include securing the distributed components involved in the software supply chain throughout the development and deployment process.

Keywords: Cloud, Cybersecurity, Containers, Orchestrations, Kubernetes, Infrastructure, Software Supply chain

Chapter 1

1. Introduction

Cloud providers managing the container services provide many ways to help secure your workloads, but does this involve protecting container workloads in different layers of stack? This stack includes different contents such as container image, the container runtime, the cluster network, and access to the data deployed in the workloads across clouds. These multiple component workflows add to the complexity of managing, securing and troubleshooting applications.

Below are some challenges that are identified with remote environments and services, which are further mapped to few categories in the next section:

- Secure source code stored locally, employees development environments, and in version control management systems.
- Build and test code on a secure infrastructure, with proper authentication and authorization to access resources.
- Scanning of container images and their runtimes for existing vulnerabilities before deploying to production environments. In addition, implement a mechanism to attest images before moving it to next stage in the pipeline.
- Managing container access and securing communications between microservices.

- Ensure applications running in the clusters across clouds are in compliance with the company policies, restricting user access to particular namespaces with proper authentication and authorization in place.
- Logging and monitoring application on a regular basis to prevent any malicious usage of cluster resources.

1.1. Problem Statement

To overcome these challenges, putting a standard security and configurations controls in place is critical. At the same time, this process should be automated to achieve the scale in the DevOps lifecycle. The idea of this research is to focus on obtaining a set of processes that address the above objectives; deep dive on securing container deployments and enforcing policies and controls for cloud native environments.

This study will lay down a set of rules that can be followed to secure the DevOps workflows for Kubernetes applications, and will cover the most critical security and reliability requirements without causing any delay in the releases and ensure operational independence. This would further detect and prevent attackers who attempt to use Kubernetes to breach the systems, by picking on the vulnerabilities.

This research would involve implementation of continuous security measures on the below given three phases of the application development lifecycle:

- Continuous Development
- Continuous Integration
- Continuous Delivery

Each of these above phases needs to be analyzed thoroughly in an organization, as a single miss on the configuration could lead to a severe data breach that can be expensive and damaging to the business. Moreover, with Kubernetes applications – management and maintenance of deployments can be crucial, hence it is very necessary to have a full proof security posture.

To start with, let's focus on each of the above development lifecycle phases and understand the key areas where security strategy can be applied. These best practices and deployment patterns can be a key to establish safeguards before moving code to the production services.

1.2 Methodologies

1.2.1 Continuous Development: This methodology comes from the agile practices, where rather than making changes in bulk, small changes to the software are delivered to the customers, as soon as it is deployed and tested. This can reap great benefits for companies, as the software is continuously improved , and proper feedback is received on a timely basis. In order to apply a security strategy to this phase, the developers should be empowered to secure code, the moment they write it. This could be as simple as, making sure their workstation is not left unattended. The paper would further detail around these best practices and how they can be implemented to ensure the security during this stage of process.

1.2.2 Continuous Integration(CI): It refers to the set of tools and processes that run on a continuous* basis to ensure that changes to a codebase are proper, safe, and accurate. A CI System often involves several components:

- A **Continuous Build** (CB) component to ensure that code changes do not break dependencies.
- A **Continuous Testing** (CT) component to ensure code changes do not alter the expected behavior of those dependencies.
- A **Continuous Deployment** (CD) component to ensure code changes can be pushed to production.

In order to ensure that risks are detected and managed in earlier stages of these processes before it gets more complex, it is very important to bake security into each of these layers. The prevention could lie anywhere between scanning container images for addressing vulnerabilities that can be introduced during the build stage, identifying runtime dependencies that can cause breaches to the system, to encrypting secrets of the source control systems.

1.2.3 Continuous Delivery: It is the ability to do automated promotion of code through deployment pipeline with quality gates to production, should the business so choose. The quality gates involves automated testing and monitoring. It automates a process of verified and secure deployment of the workload into designated environment. This stages includes integrating security operation processes. For example: RBAC rules validation in Kubernetes deployment can enable restricted access to the other resources of the application, hence ensuring only authorized users can perform actions to access resources.

To summarize, the above listed core stages of the container infrastructure pipeline needs to be implemented keeping security in mind, rather than managing security as an extended process. In this way, identifying and responding to attacks becomes easier as there are safeguards applied to

each stage of the process. This is more critical for cloud native containerized applications as the attack surface is wider, and a small misconfiguration can open the gates for the attackers to find a way into the systems.

Chapter 2

2.1 Annotated Bibliography

In the present day, security challenges that are tied to the containers and their orchestration is a major concern, and if security is not included as an inherent process for their application lifecycle, organizations would not only put their businesses at risk, but also jeopardize their customer's business. On the other side of the spectrum, according to a survey 52% of developers[6] believe that security limits the speed of innovation and time to market. Thus there is a need to have a unified approach to this problem that would empower business by combining both the objectives - that is securing applications and increase agility to adopt new changes in the system.

Kumar, Rakesh, and Rinkaj Goyal. "Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC)." Computers & Security 97 (2020): 101967. [ScienceDirect]

The authors modelled a continuous security tool to address the above mentioned issue. In the research paper, a conceptual model for automated DevSecOps using open-source software over cloud)[3] was developed that incorporated development, security and operation activities as an automation workflow and it leverages the Open Source Software(OSS) on cloud to implement this model, without compromising the time to market factor of the business. It also referred to a set of open source Infrastructure as code(IAC) tools to achieve the automation that was proposed for the

security controls. However, applying a set of security controls using an automation tool can be challenging, and not all of the companies who are managing and maintaining container infrastructure can adapt to the degree of automation that these cloud native technologies can offer. The paper further discussed on benefits of going with the open source technology stack, to make it cloud agnostic. That said, there is no 'one size fits all' solution but instead a combination of OSS tools that needs to be adapted by organizations to fulfil the objective of achieving this automation. However, this research focused on a traditional application stack, and not the microservices oriented approach, thus integrating such tools with additional operational overhead becomes difficult to adopt.

A. Sojan, R. Rajan and P. Kuvaja, "Monitoring solution for cloud-native DevSecOps," *2021*

IEEE 6th International Conference on Smart Cloud (SmartCloud), 2021, pp. 125-131, doi: 10.1109/SmartCloud52277.2021.00029. Retrieved from https://www.ieeexplore.ieee.org. To address the automation for DevSecOps for microservice architectural style, the authors derived a monitoring solutions for Cloud DevSecOps, where the model covers both infrastructure and the application level automation of the DevOps practices. This study addressed two problems – [a] Deploying a repeatable solution that can monitor cloud-native infrastructure and [b] Automation capability of the solution which is triggered by events as alerts. This introduced the idea of continuous monitoring, which is a measure to improve the overall infrastructure health, as well as application health. This solution considers infrastructure monitoring as the heart of the automation tool, however it was identified that there are limitations of the monitoring capability on the containerized workloads, container to container communications, service to service

communications etc. and thus unavailability of such DevSecOps tools and solutions for container applications made it more difficult to adopt for few organizations.

M. Zaydi and B. Nassereddine, "Devsecops practices for an agile and secure it service management", *Journal of Management Information and Decision Sciences*, vol. 23, no. 2, pp. 1-16, 2020. To conquer the challenges mentioned above, Security Principles for DevOps and cloud[7] provided a great overview of using cloud transformation to balance security activities and how organizations can keep up with the rapidly changing environments(Containers as a Service(CAAS), Serverless, etc.). The study included a baseline of Security and DevOps principles that can be applied to software lifecycle for applications running on any cloud, and any technology stack, hence ensuring no vendor lock-in limitation for this proposal. This research paper is an extension of this study[7], as it focusses on implementing the cloud and security principles to each phase of the containerized application lifecycle. Retrieved from https://www.compact.nl/pdf/C-2020-1-Sprengers.pdf

S. Kamthania, "A Novel Deep Learning RBM Based Algorithm for Securing Containers," 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), 2019, pp. 1-7, doi: 10.1109/WIECON-ECE48653.2019.9019985.

Retrieved from https://www.ieeexplore.ieee.org

With the advancement in technology, the procedures to detect vulnerabilities at an early stage are becoming more prominent. One such mechanism was discussed by the authors of this paper, where

machine learning algorithms can be leveraged to assess the container workloads. This research elaborates on how the containers can be configured at an earlier stage of development and align with the security reviews. With Boltzman Machine learning algorithm, container behavioral stats can be extracted during run-time, and the NIST security rules can be automatically applied wherever there are any security violations. In the end it creates a profile for the container that can be reviewed by the security engineers to evaluate the deployments.

O. Díaz, M. Muñoz and J. Mejía, "Responsive infrastructure with cybersecurity for automated

high availability DevSecOps processes," 2019 8th International Conference On Software Process Improvement (CIMPS), 2019, pp. 1-9, doi: 10.1109/CIMPS49236.2019.9082439. Retrieved from https://www.ieeexplore.ieee.org

The authors from Mexico discusses around the importance of **responsive infrastructure** and argues that these strategies should be incorporated within the processes of deployments as well. They mentioned how such processes can be automated and organizations can build highly available and fault tolerant systems using microservices. This is one such approach where risk management strategies were embedded in each stage of deployment for microservices.

D. B. Bose, A. Rahman and S. I. Shamim, "'Under-reported' Security Defects in Kubernetes

Manifests," 2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS), 2021, pp. 9-12, doi: 10.1109/EnCyCriS52570.2021.00009. Retrieved from https://www.ieeexplore.ieee.org

Kubernetes researchers D. B. Bose, A. Rahman and S. I. Shamim conducted a broad examination of all the security defects that have been reported for Kubernetes projects and provided a list of observations from those defects. Kubernetes as an orchestrating platform is widely used and security defects can be injected within any component of this platform. The authors pursued this objective and detected the frequency of such defects and venues where they are mostly found, whether in the configuration manifests, repositories etc. They also highlighted the unresolved defects that could be responsible for a larger data breach.

This research is crucial to the topic discussed as it gives insights to the vulnerabilities that are often neglected and can cause a significant amount of damage at alarming rates.

M. S. Islam Shamim, F. Ahamed Bhuiyan and A. Rahman, "XI Commandments of Kubernetes

Security: A Systematization of Knowledge Related to Kubernetes Security Practices," 2020 IEEE Secure Development (SecDev), 2020, pp. 58-64, doi: 10.1109/SecDev45635.2020.00025.

Retrieved from https://www.ieeexplore.ieee.org

Based on the Tesla security breach incident that occurred in 2018 associated with the Kubernetes deployments, the authors of this research report developed a set of security practices ~ 11 in total, to help practitioners protect their application's supply chain process. They found that the practices revolve around three pillars: Role based access control, Kubernetes version management and implementing network and pod security policies to prevent any malicious access to the resources,

Tanya Janca, "Securing Modern Applications and Systems," in Alice and Bob Learn Application

Security, Wiley, 2021, pp.167-191. Retrieved from https://www.ieeexplore.ieee.org

Application Security researcher Tanya provides a high-level overview of security tooling for different components, processes, and workflows through this research paper. The author reviewed studies of various security tactics and how they apply to each component of the software development lifecycle. The security of APIs is also a key part of the discussion as it acts as a medium of communication between two software and is critical in application security.

J. Mahboob and J. Coffman, "A Kubernetes CI/CD Pipeline with Asylo as a Trusted Execution

Environment Abstraction Framework," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 0529-0535, doi: 10.1109/CCWC51732.2021.9376148. Retrieved from https://www.ieeexplore.ieee.org

The authors assert that security of the finished artifact should not be the only asset to protect in a DevOps environment. The artifacts that travel through the source till the final deployment targets should be monitored and protected from any types of data breach. The paper revolves around four pillars of ensuring security, which starts with the Continuous Integration and Continuous deployment processes should have support for strong separation of duties, next the entire execution of security best practices should be automated, third the orchestration engine should be supported in different ecosystems/cloud providers and finally the deployment artifacts should be secured using a certain dedicated framework/tool for trusted environments.

Chris Binnie; Rory McCune, "DevSecOps Tooling," in Cloud Native Security , Wiley, 2021,

pp.103-104. Retrieved from https://www.ieeexplore.ieee.org

The authors from this research paper provided directions towards achieving a secure infrastructure for cloud containers, along with hands-on examples of how to mitigate attacks in certain areas of concern and isolating systems to prevent these threats affecting the tightly coupled systems. This also involved configuring various DevSecOps tools and building an automated ecosystem around your container application and the orchestrator as well.

The research series is relevant to my paper as it focusses on providing a guide to secure your cloud native technology stack at all layers of the application lifecycle. In addition, it also helps in identifying gaps in the security of current infrastructure components that could be addressed using security policies.

Chapter 3

3.1 Fusing DevOps and Container Security

This section will give an overview of the DevSecOps model applied to containerized applications in cloud and how organizations can invest in automation throughout the infrastructure and application lifecycle. The proposed solution focusses on addressing the below given high level requirements:

(A) Reducing the attack surface during the development phase of the application lifecycle, without failing to keep up with the business. This would include adapting a managed service approach for collaboration and version control.

(B) Adapt automation to speed up integration of code using cloud-native and kube-native solutions, by embedding security in the build and test infrastructure.

(C) Automate release pipelines to deploy container workloads securely, this would built upon the principle of Infrastructure as a code(IAAC).

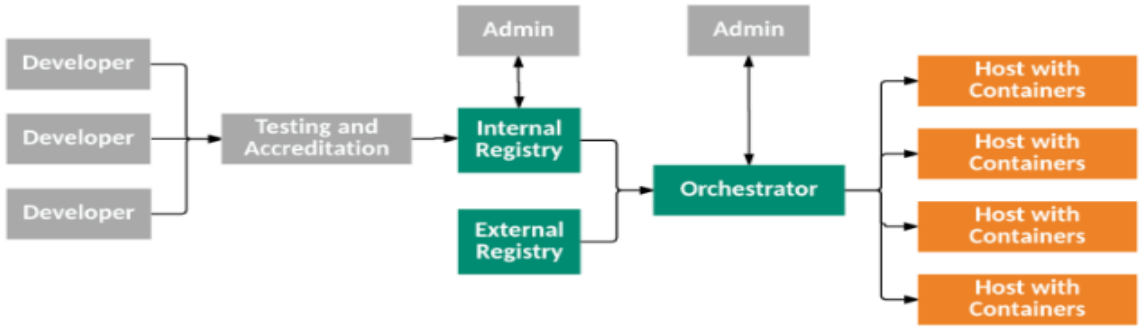
The above given specifications can be implemented using a broad ecosystem of container and kube-native solutions available, that varies from tools and different frameworks, which can integrate with your application. This model will work around building a DevSecOps pipeline that ensures the security processes are intact, when automating tasks for various stages and processes. The requirements listed above might not be complete and can accommodate additional use cases, however for the current scope of this project, this would be the base of the model for implementing continuous security and the listed requirements will be addressed in the next section.

3.1.1 Components and Controls

As two thirds of businesses are unhappy with the release speed, the goal of this research is to increase speed delivery, while maintaining high service security. This approach would integrate security guidelines in all stages of your Kubernetes deployment pipeline. The proposed solution would cover workflow for building, pushing and deploying your container application, using tools and services that would be cloud-agnostic, as well as help how you can modernize your software delivery solution.

The goal of this framework is to provide a check-list of standards, best practices and controls to prevent tampering, improve integrity, secure workloads and infrastructure in your projects, businesses or enterprises. It is how you get from safe enough to being as resilient as possible, at any link in the chain.

The below architecture reflects a simple container deployment pipeline with the different personas handlings different components of the infrastructure.



Now, risks can be introduced at any stage of this pipeline by any actor if proper controls and measures are not applied to the workflow. The proposed framework would concentrate on addressing the below risks:

- **Image Risks:** Untrusted images may introduce vulnerabilities and malware into your applications
- **Registry Risks:** Unsecured registries can allow unauthorized access or the distribution of malicious images.
- **Orchestrator Risks:** Misconfigured orchestrators can create broad attack surfaces and allow unauthorized cluster control.
- **Container Risks:** Poorly isolated containers can enable privilege escalation and lateral movement within your environment.

- **Host OS Risks:** Vulnerabilities in the host OS could compromise the security of all running containers.

3.1.2 Requirements Mappings

The reference architecture can be extended to other cloud or on-premises. Products and services adopted in the reference architecture can be substituted.

Requirement No.	Product or Services	Requirements Description	Available Products
Req 001	Source Repository	Store application and infrastructure source code	Github, GitLab, BitBucket, CodeCommit
Req 002	Build Tool	Orchestrate the continuous integration, such as tests, application & container builds, and IAC deployments	CloudBuild, Jenkins, GitLab, CircleCI, Zuul, AWS CloudBuild
Req 003	Deployment Pipeline	Manage the application and container deployments	Spinnaker, GitLab, Jenkins, CloudDeploy, CodeDeploy

Req 004	Authorization & Attestation	Sign the builds and containers to ensure only trusted containers are deployed to the Kubernetes clusters	Kritis + Grafeas + Kubernetes Admission Control
Req 005	Container & Artifact Registry	Store build artifacts, containers and perform vulnerability scan for the containers	JFrog Artifactory, Nexus, CodeArtifact, Azure Container Registry
Req 006	Kubernetes Orchestration	Automate and scale managed Kubernetes Platform	GKE, EKS, AKS, Kubernetes on Vmware, Kubernetes on Bare Metal Servers
Req 007	Logging and Monitoring	Implement Control plane logging and monitoring, including workload logs, audit and trace logs	Cloud Operations Suite, CloudWatch, CloudTrail, Azure Cloud Monitoring, Prometheus, Datadog

To adopt the approach for shift left security to secure your supply chain, the following use-cases should be followed while managing and maintaining the software delivery pipeline:

Use-Case Design

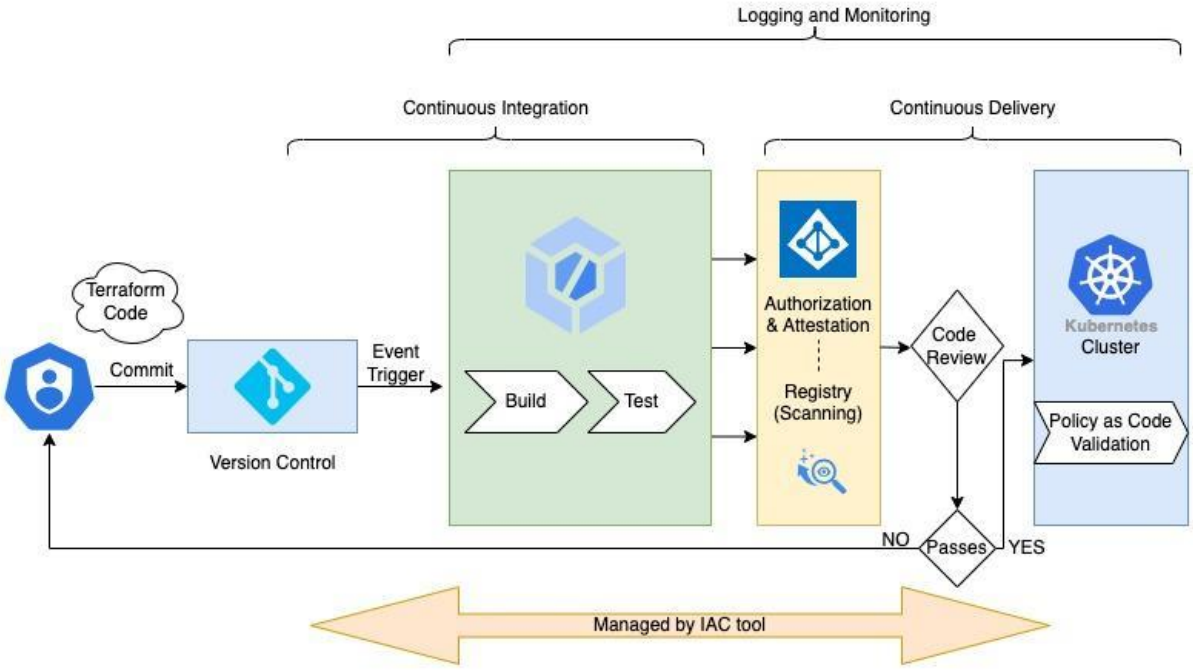
Use Case No.	Use-Cases	Description	Architecture Component
001	Provenance - Available	All builds are orchestrated in a build server, and container images are signed by an attestor who has permission to approve/disapprove images - only then the image will be deployed to the Kubernetes clusters.	Authorization & Attestation Tools such as Graeas, Sigstore(Cosign), JFrog Xray, Binary Authorization
002.	Common - Security	Leverage SIEM tools in the market to identify and prevent	Vulnerability Scan Tools

		threats at an early stage.	
003.	Common - Access	All the actions and changes made to the pipeline in the reference architecture are logged. It is recommended that users implement further monitoring & alerting leverage of the collected logs.	Logging & Monitoring Tools
004.	Common - Superusers	It is recommended locking down the console access in the production or near-production environments and deploy any changes via IaC(Infrastructure as Code) and	Terraform/other IAAC tools

		orchestrate the IaC via the pipeline.	

3.1.3 Architectural Workflow

Based on the functional requirements that were discussed in the previous section, the following diagram depicts a secure DevOps tools chain.



At the highest level, we have the following components in the basic architecture:

1. **Source Repository and development environment:** This is the hub where application code is stored. This version-control system should have restricted user access, with only authorized users having permissions to push or merge to the production application code. In order to prevent users from storing security keys in a repository, the version-control

system should be able to detect service account credentials, private keys including RSA, DSA and PGP. Developers writing code on local systems can also introduce risks, and have created challenges in securing local systems. Thus a cloud native Integrated development environment can not only help securing the systems, but also increase developer's productivity.

2. **Build tools and Continuous integration:** The code is build, tested and packaged using a continuous integration tool, and it is one of the most critical component of the DevOps tool chain. Containerized application code can have various build requirements with respect to runtime, packages, additional plugins and other connectors. In addition, the tools needs to access the source code securely, if it runs in a private network, and push the packaged container image to the registry. This requires the build tool have proper permissions to access the resources, and would need to run in a private network.
3. **Artifact Registry:** This component is responsible for storing, and managing the build artifacts such as our container images, Gradle or Maven packages. To embed security into this component, container analysis operations should be performed on the artifacts to detect vulnerability.
4. **Continuous Deployment: To automate delivery to the targets such as our Kubernetes cluster, a cloud-managed delivery tool can perform deployments in a sequence of actions, including approvals or denies to the production environments.** It also involving rollbacks, and have auditing feature embedded with the managed service
5. **Runtime Environment and Orchestration**: Within the Kubernetes security, the orchestration platform is split into the below focus areas:
    a. **Identity and Policy Enforcement**

- OPA (Open Policy Agent): A powerful policy engine that can be used to enforce complex security rules across your Kubernetes cluster. Use cases include fine-grained RBAC, pod security policies, and more.
- Kubernetes RBAC: Built-in role-based access control system. Integrate with your preferred identity provider (LDAP, OIDC, etc.)
- Commercial Tools: Many cloud security solutions (Prisma Cloud, Sysdig, etc.) offer Kubernetes-specific policy enforcement and auditing features.

b. Kubernetes Hardening

- CIS Benchmarks: Provide detailed hardening recommendations for Kubernetes.
- kubescape: Open-source tool that scans your cluster against CIS and other security benchmarks.
- kube-bench: Another popular open-source tool for checking cluster configuration against CIS benchmarks.

c. Image Inventory and Vulnerability Scanning

- Container Registry Scanning: GCP's Vulnerability Scanning, AWS ECR scanning, etc. These are often a good starting point.
- Clair, Trivy: Open-source image scanners with extensive CVE databases.
- Snyk, Anchore: Commercial tools offering more in-depth analysis and remediation guidance.

d. Data Protection and Secrets Management

- Kubernetes Secrets: Built-in, but use with caution. Limited encryption options at rest.
- Vault (HashiCorp): Popular secrets management solution, offers dynamic secrets and strong encryption.

- CyberArk: Commercial solution with a focus on privileged access and secrets management across environments.

e. Metadata Concealment

- NetworkPolicy (Kubernetes): Limit pod-to-pod traffic using network policies, reducing the attack surface.
- Service Meshes (Istio, Linkerd): Can provide fine-grained traffic control, encryption, and potentially metadata filtering.

Based on the above security onboarding checklist and requirements, these structured tasks can be followed to deploy your infrastructure and workloads securely. A managed service provider can offer all the features listed above in an automated way.

3.1.4 Design Test Cases

Test Case Number: TC1

Revision: Rev. 1

Author: Vishakha Sadhwani

Date Conducted: Dec 01, 2022

Test Conductor: Vishakha Sadhwani

Customer Representative: Vishakha Sadhwani

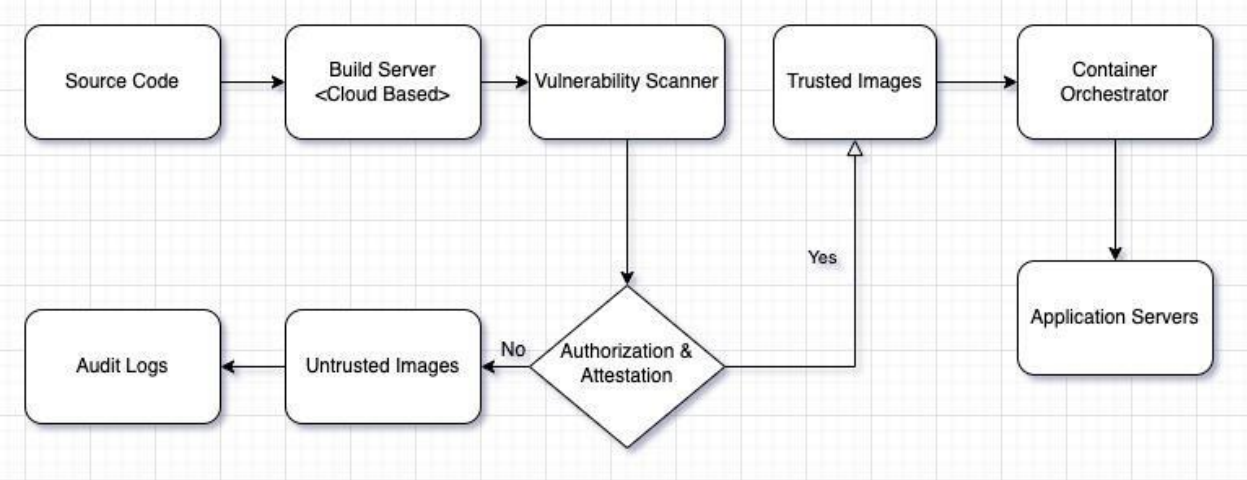
Description: This Test Case will test the functionality of attesting container images automatically to verify and validate certain specifications that are approved before deploying the image to the target Kubernetes cluster. The related use case for this Test case is UC-001

Pre-Test Setup:

1. Ensure the pipeline is configured with the deployment components, such as source and destination repositories, build server, deploy service and target staging and production kubernetes clusters.

2. A sample application is deployed to the Kubernetes cluster.

Use Case: 001, 002, 003 Flow: Main Flow



User Action	Expected Results	P/F	Comments
A developer commits changes to the main branch	An authorized user has successfully pushed code to the main branch	P	Req 001
Build triggers on commit to branch and image is built	The tool started building the code with secure packages, runtimes, and unit tests.	P	Req 002
The image is pushed to registry	Verify new image is available in the artifact registry	P	
The image is scanned by vulnerability scanner	The tool scanned the image for vulnerabilities, and none was found	P	Req 003, 004, 005
Detects non-critical vulnerabilities	The tool scanned the image for vulnerabilities, and none was found	P	Req 004, 005
An authorized user validates the results and ensures the image is safe to deploy to the	The image was verified by the authorized QA (Quality Assurance) team	P	

environment through attestations			
An authorized user updates the deployment manifest with the new image and deploys it to production cluster	The app was updated with the new image by an authorized cluster admin user	P	Req 006
Post-conditions: The cluster was updated with the latest changes made to code, and the application container's health is being monitored.			

Test Case Number: TC2

Revision: Rev. 1

Author: Vishakha Sadhwani

Date Conducted: Dec 01, 2022

Test Conductor: Vishakha Sadhwani

Customer Representative: Vishakha Sadhwani

Description: This Test Case will test if only authenticated and authorized users are accessing the resources in the pipeline, including Kubernetes cluster resources.

The related use case for this Test case is UC-001, 002

Pre-Test Setup:

1. Ensure the pipeline is configured with the deployment components, such as source and destination repositories, build server, deploy service and target staging and production Kubernetes clusters.

2. A sample application is deployed to the Kubernetes cluster.

3. Logging and Monitoring has been enabled to track the errors.

Use Case: 001, 002, 003 Flow: Main Flow

User Action	Expected Results	P/F	Comment
A user signs in to cloud console to access the project resources	Login page is displayed for the cloud console	P	
User needs to login using their web identity(userinfo.email) to be authenticated	User enters their email and password in the login page to access the console	P	
The user has successfully logged in to the cloud console	Verify user can view the resources based on the permissions associated to their user identity	P	
User can access kubernetes cluster resources by retrieving an OAuth access token	Retrieve kubernetes cluster credentials for a specific user	P	Req 006, 007
A user is authorized to perform cluster actions through RBAC (Role based access control)	RBAC roles are established with fine-grained permissions and associated with the user	P	Req 006
A user can run/manage workloads in authorized kubernetes namespaces	Users can run workloads on the authorized clusters and access allocated resources.	P	Req 006, 007
Post-conditions: User's activity can be audited through logging and monitoring, hence any unauthorized access can back			

tracked and admins are alerted to act on.

Test Case Number: TC3

Revision: Rev. 1

Author: Vishakha Sadhwani

Date Conducted: Dec 01, 2022

Test Conductor: Vishakha Sadhwani

Customer Representative: Vishakha Sadhwani

Description: This Test Case will test the functionality of the pipeline towards untrusted images that have been detected through the scanner and how the activity is logged and remediated.

The related use case for this Test case is UC-001

Pre-Test Setup:

1. Ensure the pipeline is configured with the deployment components, such as source and destination repositories, build server, deploy service and target staging and production kubernetes clusters.

2. A sample application is deployed to the kubernetes cluster.

3. Logging and Monitoring has been enabled to track the errors.

Use Case: 001, 002, 003

Flow: Main Flow

User Action	Expected Results	P/F	Comment

A developer commits changes to the main branch	An authorized user has successfully pushed code to the main branch	P	Req 001
Build triggers on commit to branch and image is built	The tool started building the code with secure packages, runtimes, and unit tests.	P	Req 002
The image is pushed to registry	Verify new image is available in the artifact registry	P	
The image is scanned by vulnerability scanner	The tool scanned the image for vulnerabilities, and **critical** vulnerabilities were found	P	Req 003, 004, 005
Detects critical vulnerabilities	The tool scanned the image for vulnerabilities, and the user ensured that the image cannot be trusted	P	
An authorized user validates the results and ensures the image is safe to deploy to the environment through attestations	The vulnerabilities detected by the scanner was logged for audit purposes and the authorized engineer can ensure to remediate those with proper testing	P	
An authorized user cancels the deployment and roll back the source code to previous version	The authorization tool verified that the app did not pass it unit tests and hence is not built using a specific set of systems, thus the code change would not progress to production systems	F	Req 001
Post-conditions: The application update did not pass the required tests and includes some critical vulnerabilities; hence this change will not progress until these introduced vulnerabilities are fixed.			

Test Case Number: TC4

Revision: Rev. 1

Author: Vishakha Sadhwani

Date Conducted: Dec 01, 2022

Test Conductor: Vishakha Sadhwani

Customer Representative: Vishakha Sadhwani

Description: This Test Case will test if all the resources that are provisioned using an (Infrastructure as a code) tool, are managed, and maintained by the IAAC (Infrastructure as a code) such as Terraform

The related use case for this Test case is UC-004

Pre-Test Setup:

1. The supported resources are planned as the infrastructure components and formulated per the IAAC tool - terraform3. Logging and Monitoring has been enabled to track the errors. Use Case: 001, 002, 003

Flow: Main Flow

User Action	Expected Results	P/F	Comment
User prepares a script to provision all the infrastructure resources through terraform	An admin user creates a script using the resource specifications	P	
User validates the script to confirm if the terraform tool is implementing	Through terraform cli, user validates the script and the expected resource configurations	P	

resources per the security best practices			
User executes the script and resources are deployed	Through terraform cli, user applies the terraform script to deploy to specific cloud provider	P	
To track the resource status, terraform tool stores the state of each resource locally	The user verifies and validates the state of the resources are stored locally	P	
User creates a backup of the state file to a remote source such cloud storage buckets	The user verifies the backup files are stored in cloud storage bucket	P	Req 007
User updates the resources through the terraform script	User makes changes to the infrastructure through the script	P	Req 007
Post-conditions: Users are only allowed to update infrastructure through the IAAC tool and terraform tool has the required permissions to perform all the actions to the cloud resources.			

Chapter 4

4.1 Results and Conclusion

The current pipeline performs the following operations:

- Applies the security checks in different layers of the stack
    - Source Repository
    - Build service
    - Authorization and Attestation Service
    - Kubernetes Platform
    - Audit Logging & Monitoring
- Leveraged a terraform script that builds the entire environment for this setup

- The terraform script is currently supporting a CI/CD pipeline implementation only. The steps automated the provisioning process, and solves the below problems:
    - Easy instantiation of resources and builds a dependency tree
    - Tracks the resource statuses in state files, so in case if any resource is accidently deleted by any user, the IAC tool would trigger an event to rebuild the resource
    - Ease to deploy in multiple environments such as Dev, QA, UAT, Production etc
    - Ease in destroying/cleaning up resources all at once

4.2 Future Work

- The current setup supports implementation only on concepts, hence there can future work done on creating scripts for cloud platforms that will deploy these security baselines in an automated way.
- In this project, I have not focused on the cost of services involved in implementing the security checklist. This can be a good next step to dive into, along with developing a cost optimized setup.
- There are a lot of open-source SIEM tools currently available in the market, for future work, an integration with an incidence management tool can be particularly useful.

References:

1. Kumar, Rakesh, and Rinkaj Goyal. "Modeling continuous security: A conceptual model for automated DevSecOps using open-source software over cloud (ADOC)." Computers & Security 97 (2020): 101967. [ScienceDirect]
2. A. Sojan, R. Rajan and P. Kuvaja, "Monitoring solution for cloud-native DevSecOps," *2021 IEEE 6th International Conference on Smart Cloud (SmartCloud)*, 2021, pp. 125-131, doi: 10.1109/SmartCloud52277.2021.00029.

3. M. Zaydi and B. Nassereddine, "Devsecops practices for an agile and secure it service management", *Journal of Management Information and Decision Sciences*, vol. 23, no. 2, pp. 1-16, 2020.
4. S. Kamthania, "A Novel Deep Learning RBM Based Algorithm for Securing Containers," 2019 IEEE International WIE Conference on Electrical and Computer Engineering (WIECON-ECE), 2019, pp. 1-7, doi: 10.1109/WIECON-ECE48653.2019.9019985. Retrieved from https://www.ieeexplore.ieee.org
5. O. Díaz, M. Muñoz and J. Mejía, "Responsive infrastructure with cybersecurity for automated high availability DevSecOps processes," 2019 8th International Conference On Software Process Improvement (CIMPS), 2019, pp. 1-9, doi: 10.1109/CIMPS49236.2019.9082439. Retrieved from https://www.ieeexplore.ieee.org
6. D. B. Bose, A. Rahman and S. I. Shamim, "'Under-reported' Security Defects in Kubernetes Manifests," 2021 IEEE/ACM 2nd International Workshop on Engineering and Cybersecurity of Critical Systems (EnCyCriS), 2021, pp. 9-12, doi: 10.1109/EnCyCriS52570.2021.00009. Retrieved from https://www.ieeexplore.ieee.org
7. M. S. Islam Shamim, F. Ahamed Bhuiyan and A. Rahman, "XI Commandments of Kubernetes Security: A Systematization of Knowledge Related to Kubernetes Security Practices," 2020 IEEE Secure Development (SecDev), 2020, pp. 58-64, doi: 10.1109/SecDev45635.2020.00025. Retrieved from https://www.ieeexplore.ieee.org
8. Tanya Janca, "Securing Modern Applications and Systems," in Alice and Bob Learn Application Security, Wiley, 2021, pp.167-191. Retrieved from https://www.ieeexplore.ieee.org

9. J. Mahboob and J. Coffman, "A Kubernetes CI/CD Pipeline with Asylo as a Trusted Execution Environment Abstraction Framework," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 0529-0535, doi: 10.1109/CCWC51732.2021.9376148. Retrieved from https://www.ieeexplore.ieee.org
10. Chris Binnie; Rory McCune, "DevSecOps Tooling," in Cloud Native Security , Wiley, 2021, pp.103-104. Retrieved from https://www.ieeexplore.ieee.org
11. Dark Reading Staff - Strained Relationships Hinder DevSecOps Innovation
12. W. van der Houven, Security principles for devops and cloud
13. Kirsten Newcomer Why DevSecOps Is Critical for Containers and Kubernetes
14. Robert Lemos Software-Container Supply Chain Sees Spike in Attacks