KUBERNETES SECURITY FOR DEVOPS PIPELINES

Security is among the primary considerations in any application development. Security standards and practices should be integrated into all aspects of software development, from infrastructure and databases to maintenance and management.

Kubernetes supports plenty of security configurations that can be used to secure clusters and underlying pods. In this article, we will see what the Kubernetes security best practices are and how to integrate them into DevOps.

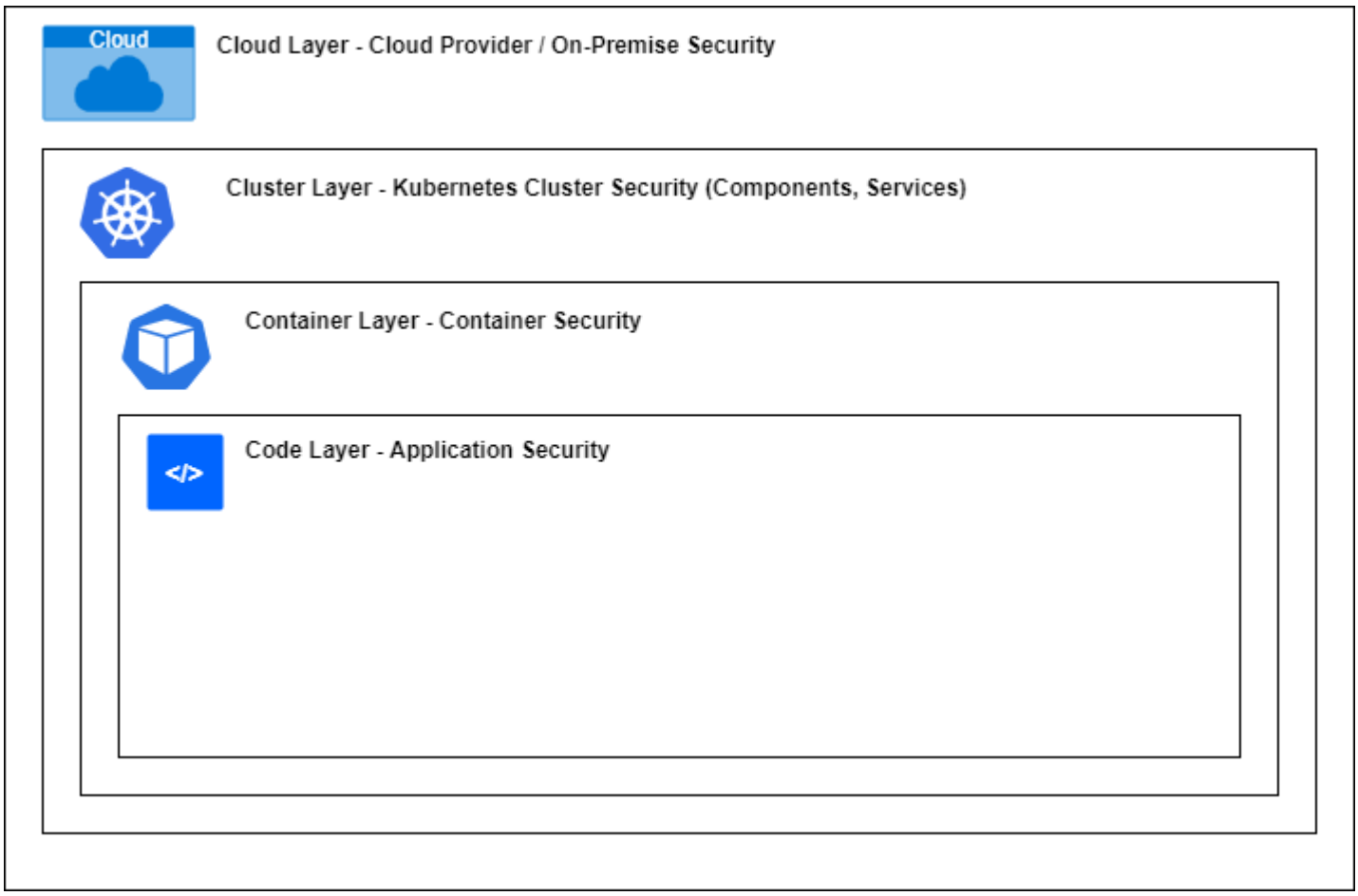
(This article is part of our Kubernetes Guide. Use the right-hand menu to navigate.)

Cloud-native security

Let's begin with a look at the cloud-native security architecture when identifying the base of Kubernetes security. The 4Cs in Cloud-native security refers to the four security layers:

- Cloud
- Clusters
- Containers
- Code

Each layer is built on top of the previous layer providing maximum security to a Kubernetes cluster:



- **Cloud.** This layer represents the underlying physical infrastructure. You may deploy your Kubernetes application either on-premise hardware or on a third-party cloud provider, but you need to follow the relevant security best practices to secure all the infrastructure.
- **Cluster.** A Kubernetes cluster should be securely configured with cluster services like Kubernetes API and all the applications within the cluster. Securing applications is even more important when dealing with microservices since a minor vulnerability in a single service can affect the whole cluster.
- **Containers.** Build your containers using the smallest possible image by enforcing strict user privileges and removing all unnecessary libraries and features. Additionally, all containers must be scanned regularly to identify any vulnerabilities.
- **Code.** The application code can be the largest attack surface due to production bugs and failure to follow security practices during the development, deployment, and management. Simple actions like removing unused ports, encrypting data transfers, regular testing, and scanning can mitigate many security issues in the application code.

Kubernetes security best practices

Let's have a look at some best practices and recommendations for creating a secure Kubernetes environment.

Cloud security best practices (Infrastructure)

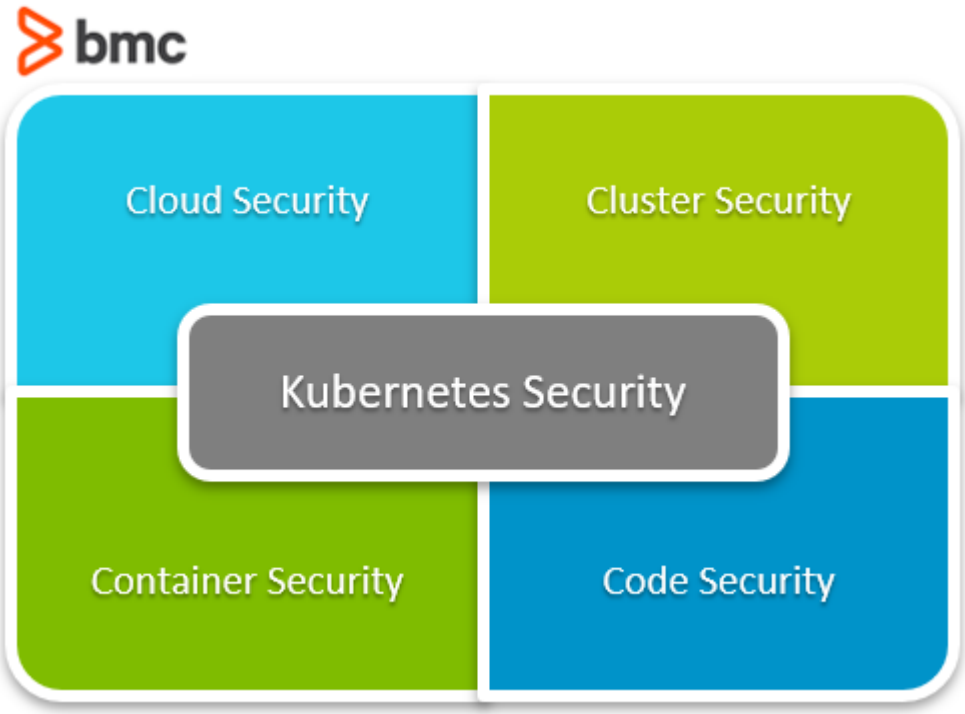
- **Control panel access.** Only a selected set of users from an authorized location (whitelisted IP) should be allowed to interact with the Kubernetes control panel. All public access should be

disabled.

- **Node access.** Access to Kubernetes nodes should be controlled by only accepting connections from Kubernetes services like NodePort or LoadBalancer restricted to specific ports and protocols. Public access to nodes should also be discouraged if possible.
- **Cloud Provider API.** When interacting with cloud provider APIs through Kubernetes, only assign roles and policies that limit the privileges to the core functions required by Kubernetes, such as provisioning new nodes.
- **Kubernetes etcd.** The access to datastore should be limited to the Kubernetes control panel and be configured with TLS. Besides, the datastore should be encrypted for additional security.

Cluster security best practices

- **Role-based access control (RBAC).** Access for all services should be limited to authorized users and roles with specific IP addresses.
- **Pod security policies.** Enforce standardized pod security policies across the cluster environment.
- **Network security policies.** Enforce network policies for all network-related resources to ensure data security.
- **Data encryption.** By default, encrypt all possible data storage from etcd to volumes where sensitive data are stored.



Container security best practices

- **Vulnerability scanning.** Scan and identify any security issues related to the container images and address them before building the container.
- **Image signing.** Enforce image signing so that the cluster can only deploy signed trusted images.
- **User control.** All privileged users should be removed, and only users with limited privileges should be included in the container (user vs root).
- **Dependency management.** Any unnecessary features or software dependencies should be removed from the base image itself.

Code security best practices

- **Encrypted communication.** Integrate encryption to all the network traffic to minimize any data leaks.
- **Application access.** Expose a limited number of application endpoints and ports and enforce authentication to reduce the attack surface.
- **Dependency management.** As with the container base image, strip out all unnecessary third-party frameworks and libraries to reduce the risk of third-party vulnerabilities affecting the application.
- **Development methodology.** Security best practices should be integrated into the SDLC as part of the DevSecOps process with regular vulnerability scanning and code reviews. This would help developers to identify and fix any security issues before the product is released for production.

Integrating Kubernetes security to a DevOps pipeline

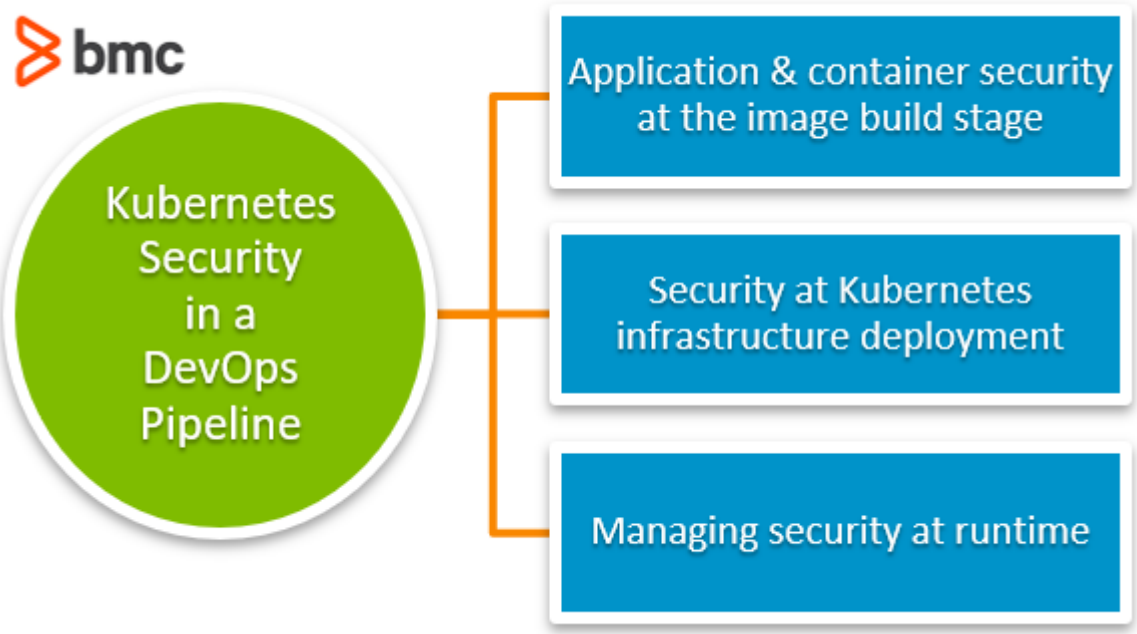
There are two major considerations that any organization will face when implementing Kubernetes security:

- **Self-configuration.** Kubernetes consists of several features to enforce security best practices. Importantly, none of them are configured by default, so the cluster administrator must manually configure them in the cluster.
- **Lack of knowledge.** It requires an in-depth understanding of the Kubernetes environment and the application to properly configure all the security features, policies, and best practices.

The best option to overcome the above issues would be integrating Kubernetes security best practices in a DevOps pipeline. It enables developers to both:

- Get a broader view of the entire application architecture
- Gradually incorporate security practices into a Kubernetes environment with incremental changes

We can boil down the Kubernetes DevOps pipeline into three distinct sections to better implement the security practices.



Let's have a look at each.

Application & container security at the image build stage

There are a number of application security best practices, such as:

- Data encryptions
- User privileges
- Token management
- Threat assessments
- Code reviews

These should be organically integrated into the development, from the initial stage of application development. Since the application is the core component of the container, an insecure application can expose the whole production Kubernetes cluster to attacks—even if all the other aspects are secure.

When building the container image, use an updated base image without any security vulnerabilities

and strip out any unnecessary components or features of it.

For example, suppose we can use a smaller base image such as alpine Linux instead of Ubuntu without affecting the functionality or performance of the application. In that case, we should always go with the smaller base image. It both:

- Reduces the attack surface
- Creates a much more lightweight container

Then we should carefully go through all application dependencies to identify vulnerabilities using a tool like synk and address them.

Another concern is the ports exposed via the container. We should only expose the necessary ports and require authentication for all requests. In the end, we can build a container image that is ready to be deployed in Kubernetes.

Another best practice is to conduct a penetration test of the application in a staging environment and sign the image before pushing it to the production cluster.

Security at Kubernetes infrastructure deployments

You must have in-depth knowledge of Kubernetes to properly configure the cluster when deploying a Kubernetes cluster. As a rule of thumb, we should try to avoid using any default configurations. We'll also need to:

- Scope user permissions in the cluster
- Set up proper network connectivity with ingress and load balancers to control network traffic within the cluster
- Create network segmentation to provide maximum physical security

Additionally, proper networking will eliminate the need for nodes to be exposed directly to the internet.

Having configured pod policies configured, we can ensure that only pods which comply with the required policy will be created, and no rogue pods will be created. Network policies help you to control:

- The traffic flow to pods
- How a pod can communicate with other network objects, pods, etc.

Another important consideration is cloud security. Even though we configure a secure Kubernetes cluster, our applications can be exposed to attacks if the underlying infrastructure or cloud provider is insecure. Therefore, administrators should secure the underlying infrastructure with proper access controls and authentication mechanisms before deploying infrastructure so that only authorized users can modify infrastructure.

Finally, the administrators need to enforce images registries so that only signed images from authorized container registries can be deployed in the cluster.

It is also essential to configure continuous monitoring and vulnerability scanning for both containers and clusters. It will act as an early warning system to identify issues and fix them. A secure container image created in the build stage combined with a secure cluster environment provides a solid

foundation for Kubernetes security.

Managing security at runtime

This final section is about managing security at runtime.

Users need to maintain security practices even after the deployment, or else the application and the cluster will become vulnerable over time. Furthermore, you'll need to keep track of all the security layers from the code to the cloud layer at runtime.

The production bugs of an application that are discovered at the production can also introduce vulnerabilities. So, a proper process should be in place to quickly deploy a fix for the bug. Besides, clusters can quickly become complex structures with multiple applications and services running in them. Thus, to maintain the security of these complex clusters, you'll need to have:

- Strict advanced network control policies
- Multi-tenant support
- Properly configured firewalls
- Encryption standards

In case of a vulnerability, the affected pod(s) should be quickly isolated and removed while new replacement pods with patched vulnerabilities are created

Lastly, you'll need to rely on a proper health monitoring and log management solution to identify all these issues. All in all, these security practices help users to ensure Kubernetes security at runtime.

Kubernetes security has no stopping point

Kubernetes security is a vast subject. In short, the process of securing an application starts from inception and continues until the EOL of the application.

Securing Kubernetes goes hand in hand with securing the application since application vulnerabilities can affect the cluster. As I've outlined here, the best approach is to include security practices in the DevOps process, considering all aspects of the Kubernetes cluster and application to maintain the security.

Related reading

- BMC DevOps Blog
- Kubernetes Best Practices for Enhanced Cluster Efficiency
- Kubernetes Multi-Clusters: How & Why To Use Them
- Kubernetes vs Docker Swarm: Comparing Container Orchestration Tools
- SecOps vs DevSecOps: What's The Difference?