Overview of Cloud Native Security

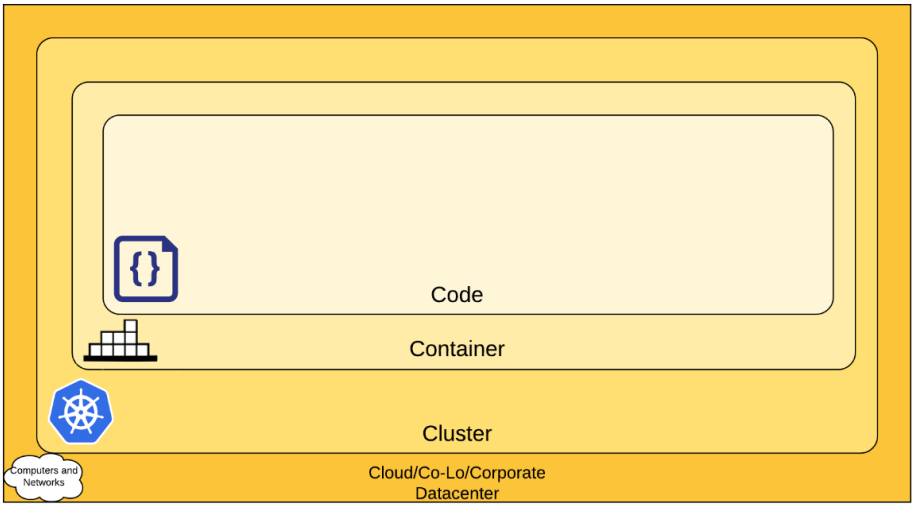
This overview defines a model for thinking about Kubernetes security in the context of Cloud Native security.

Warning: This container security model provides suggestions, not proven information security policies.

The 4C's of Cloud Native security

You can think about security in layers. The 4C's of Cloud Native security are Cloud, Clusters, Containers, and Code.

Note: This layered approach augments the defense in depth computing approach to security, which is widely regarded as a best practice for securing software systems.



The 4C's of Cloud Native Security

Each layer of the Cloud Native security model builds upon the next outermost layer. The Code layer benefits from strong base (Cloud, Cluster, Container) security layers. You cannot safeguard against poor security standards in the base layers by addressing security at the Code level.

Cloud

In many ways, the Cloud (or co-located servers, or the corporate datacenter) is the trusted computing base of a Kubernetes cluster. If the Cloud layer is vulnerable (or configured in a vulnerable way) then there is no guarantee that the components built on top of this base are secure. Each cloud provider makes security recommendations for running workloads securely in their environment.

Cloud provider security

If you are running a Kubernetes cluster on your own hardware or a different cloud provider, consult your documentation for security best practices. Here are links to some of the popular cloud providers' security documentation:

IaaS Provider	Link
Alibaba Cloud	https://www.alibabacloud.com/trust-center
Amazon Web Services	https://aws.amazon.com/security
Google Cloud Platform	https://cloud.google.com/security
Huawei Cloud	https://www.huaweicloud.com/securecenter/overallsafety
IBM Cloud	https://www.ibm.com/cloud/security
Microsoft Azure	https://docs.microsoft.com/en-us/azure/security/azure-security
Oracle Cloud Infrastructure	https://www.oracle.com/security
VMware vSphere	https://www.vmware.com/security/hardening-guides

Infrastructure security

Suggestions for securing your infrastructure in a Kubernetes cluster:

Area of Concern for

Edit this page
Create child page
Create an issue
Print entire section

Kubernetes Infrastructure	Recommendation
Network access to API Server (Control plane)	All access to the Kubernetes control plane is not allowed publicly on the internet and is controlled by network access control lists restricted to the set of IP addresses needed to administer the cluster.
Network access to Nodes (nodes)	Nodes should be configured to *only* accept connections (via network access control lists) from the control plane on the specified ports, and accept connections for services in Kubernetes of type NodePort and LoadBalancer. If possible, these nodes should not be exposed on the public internet entirely.
Kubernetes access to Cloud Provider API	Each cloud provider needs to grant a different set of permissions to the Kubernetes control plane and nodes. It is best to provide the cluster with cloud provider access that follows the principle of least privilege for the resources it needs to administer. The Kops documentation provides information about IAM policies and roles.
Access to etcd	Access to etcd (the datastore of Kubernetes) should be limited to the control plane only. Depending on your configuration, you should attempt to use etcd over TLS. More information can be found in the etcd documentation.
etcd Encryption	Wherever possible it's a good practice to encrypt all storage at rest, and since etcd holds the state of the entire cluster (including Secrets) its disk should especially be encrypted at rest.

Cluster

There are two areas of concern for securing Kubernetes:

- Securing the cluster components that are configurable
- Securing the applications which run in the cluster

Components of the Cluster

If you want to protect your cluster from accidental or malicious access and adopt good information practices, read and follow the advice about securing your cluster.

Components in the cluster (your application)

Depending on the attack surface of your application, you may want to focus on specific aspects of security. For example: If you are running a service (Service A) that is critical in a chain of other resources and a separate workload (Service B) which is vulnerable to a resource exhaustion attack, then the risk of compromising Service A is high if you do not limit the resources of Service B. The following table lists areas of security concerns and recommendations for securing workloads running in Kubernetes:

Area of Concern for Workload Security	Recommendation
RBAC Authorization (Access to the Kubernetes API)	https://kubernetes.io/docs/reference/access-authn-authz/rbac/
Authentication	https://kubernetes.io/docs/concepts/security/controlling-access/
Application secrets management (and encrypting them in etcd at rest)	https://kubernetes.io/docs/concepts/configuration/secret/ https://kubernetes.io/docs/tasks/administer-cluster/encrypt-data/
Ensuring that pods meet defined Pod Security Standards	https://kubernetes.io/docs/concepts/security/pod-security-standards/#policy-instantiation
Quality of Service (and Cluster resource management)	https://kubernetes.io/docs/tasks/configure-pod-container/quality-service-pod/
Network Policies	https://kubernetes.io/docs/concepts/services-networking/network-policies/
TLS for Kubernetes Ingress	https://kubernetes.io/docs/concepts/services-networking/ingress/#tls

Container

Container security is outside the scope of this guide. Here are general recommendations and links to explore this topic:

Area of Concern for Containers	Recommendation
Container Vulnerability Scanning and OS Dependency Security	As part of an image build step, you should scan your containers for known vulnerabilities.
Image Signing and Enforcement	Sign container images to maintain a system of trust for the content of your containers.
Disallow privileged users	When constructing containers, consult your documentation for how to create users inside of the containers that have the least level of operating system privilege necessary in order to carry out the goal of the container.
Use container runtime with stronger isolation	Select container runtime classes that provide stronger isolation.

Code

Application code is one of the primary attack surfaces over which you have the most control. While securing application code is outside of the Kubernetes security topic, here are recommendations to protect application code:

Code security

Area of Concern for Code	Recommendation
Access over TLS only	If your code needs to communicate by TCP, perform a TLS handshake with the client ahead of time. With the exception of a few cases, encrypt everything in transit. Going one step further, it's a good idea to encrypt network traffic between services. This can be done through a process known as mutual TLS authentication or mTLS which performs a two sided verification of communication between two certificate holding services.
Limiting port ranges of communication	This recommendation may be a bit self-explanatory, but wherever possible you should only expose the ports on your service that are absolutely essential for communication or metric gathering.
3rd Party Dependency Security	It is a good practice to regularly scan your application's third party libraries for known security vulnerabilities. Each programming language has a tool for performing this check automatically.

Static Code Analysis	Most languages provide a way for a snippet of code to be analyzed for any potentially unsafe coding practices. Whenever possible you should perform checks using automated tooling that can scan codebases for common security errors. Some of the tools can be found at: https://owasp.org/www-community/Source_Code_Analysis_Tools
Dynamic probing attacks	There are a few automated tools that you can run against your service to try some of the well known service attacks. These include SQL injection, CSRF, and XSS. One of the most popular dynamic analysis tools is the OWASP Zed Attack proxy tool.

What's next

Learn about related Kubernetes security topics:

- Pod security standards
- Network policies for Pods
- Controlling Access to the Kubernetes API
- Securing your cluster
- Data encryption in transit for the control plane
- Data encryption at rest
- Secrets in Kubernetes
- Runtime class

Feedback

Was this page helpful?

Yes No

Last modified September 03, 2022 at 10:37 PM PST: Fix typo and consistency: /security/overview.md (922aed0bf8)