



Welcome to

PROSA Kubernetes Security

A holistic approach to running K8s in production

Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

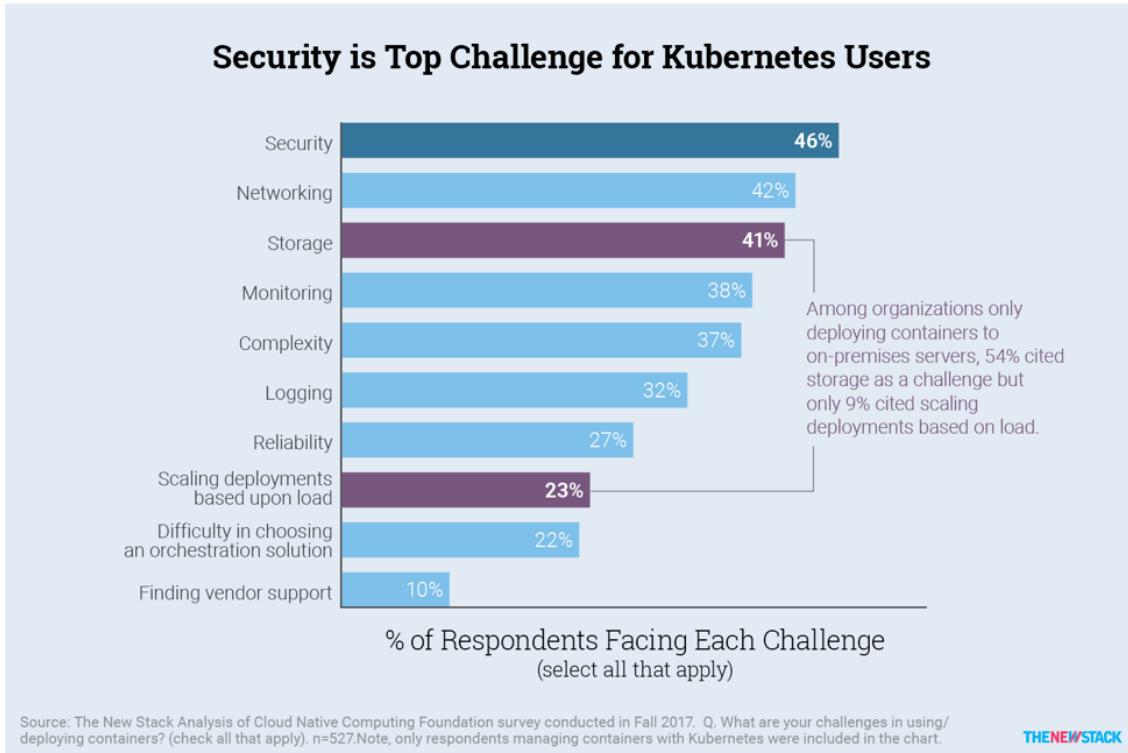
Slides are available as PDF, kramse@Github 
kubernetes-security-short.tex in the repo security-courses

Goals for today



- Create an understanding of Kubernetes attack surface
- Get an idea of the work needed to create a secure Kuberntes deployment
- Present some parts of the solution – including specific tools 🔧
- Point you towards resources, so you can get started with Kubernetes more securely
- Lay out a plan for a modern featureful reasonably secure K8s bare-metal install

Security is a Concern in Kubernetes Deployments



Source: <https://thenewstack.io/top-challenges-kubernetes-users-face-deployment/>

I come from Security – Security engineering a job role



On any given day, you may be challenged to:

- Create new ways to solve existing production security issues
- Configure and install firewalls and intrusion detection systems
- Perform vulnerability testing, risk analyses and security assessments
- Develop automation scripts to handle and track incidents
- Investigate intrusion incidents, conduct forensic investigations and incident responses
- Collaborate with colleagues on authentication, authorization and encryption solutions
- Evaluate new technologies and processes that enhance security capabilities
- Test security solutions using industry standard analysis criteria
- Deliver technical reports and formal papers on test findings
- Respond to information security issues during each stage of a project's lifecycle
- Supervise changes in software, hardware, facilities, telecommunications and user needs
- Define, implement and maintain corporate security policies
- Analyze and advise on new security technologies and program conformance
- Recommend modifications in legal, technical and regulatory areas that affect IT security

Source: <https://www.cyberdegrees.org/jobs/security-engineer/>

also https://en.wikipedia.org/wiki/Security_engineering

My Intentions



I suspect you want to work with Kubernetes, maybe you already do, but:

- You are responsible for all of it
- You don't have the resources, knowledge, time, etc. for getting to know it all
- You have to create an architecture, as well as implement it
- ... and monitor it, ... and secure it

My intention is to be your sparring partner today, list all the parts I think must be considered.

Disclaimer: I am not an expert in **EVERYTHING Kubernetes**



I am not an EXPERT

– but I have read a lot and watched youtube videos. ☺

Kubernetes is such a big ecosystem

The list is long, and an incomplete list contains:

- Kubernetes core components – software written by the project RBAC etc.
- Kubernetes supported container technologies, docker etc.
- Kubernetes dependencies – etcd software used for storing data
- Operating systems – and that lies below, storage
- Kubernetes networking providers – multiple exist, which one to choose – like Cilium
- Monitoring solutions – logging solutions

and on top all the applications in and around Kubernetes – NGINX, PostgreSQL, ...

Why are we here today then?!



We are here to get an overview:

- Identify what we *need* for a Kubernetes production environment
- Consider more parts of the picture than *just Kubernetes core*
- Find a solution that works, with some example technologies
- Provide a kind of check list for your deployment

In general I want to take a more holistic approach to
Kubernetes Security which I hope can help you



More Disclaimers



- Disclaimer: **We cover security today**

We will not cover all parts of K8s - only the ones that we need for security, as few as possible.

This slideshow is not a replacement for the official docs, and we also recommend multiple resources books, articles and papers detailing specific parts

- Disclaimer: **Deprecation**

Things are built, and later deprecated. Even while preparing this a few things were deprecated! Core things even change from time to time, which is good. In general I don't trust older references, but verify with newest official docs

- Disclaimer: **New features**

New and very useful features are added with every new version. If you have an older setup which is not updated, you might not have the same features. New features can also have interesting bugs, and may not work as intended right away. For this presentation I recommend some features that are newer

- Disclaimer: **Single tenant only**

Running a K8s cluster for your organization is hard enough, I don't claim I could run a production multi-tenancy setup or public cloud

Materials – where to start



- This presentation – slides for today, start here
- *Kubernetes: Up and Running*, 3rd Edition, Brendan Burns, Joe Beda, Kelsey Hightower, August 2022
Introduces containers and Kubernetes, books in 3rd ed also has been fixed and updated
- *Container Security*, (CS) Liz Rice, O'Reilly, Apr 2020
A classic text about containers and a must read for everyone
- *Networking and Kubernetes: A Layered Approach*, James Strong, Vallery Lancey, O'Reilly, 2021
K8s uses a lot of intricate networking – great for understanding this more
- *Learn Kubernetes Security* (LKS), Kaizhe Huang , Pranjal Jumde, Packt, July 2020
This is a very complete and detailed view at K8s security, covering many many parts
- Advanced security practitioners will enjoy security focused books like:
Hacking Kubernetes: Threat-Driven Analysis and Defense, Andrew Martin, Michael Hausenblas, O'Reilly, October 2021
- *Kubernetes Best Practices: Blueprints for Building Successful Applications on Kubernetes*, Brendan Burns, Eddie Villalba, Dave Streb and Lachlan Evenson, O'Reilly 2020



Creating a lab is not expensive

I use ordinary IT-equipment like switches, PCs and servers, with some additional 10Gbps network devices – nothing really special running Debian Linux

The main purposes for showing this box, is to show

- These are standard network devices, bought the Arista 7150 24-port 10G used on ebay
- This specific switch does VXLAN VTEP and BGP, which I use in my labs
- OpenBSD which I use for routing also has built-in VXLAN and BGP



You should have similar (or better) devices in your production network, and they can be configured to do a LOT more than you use them for right now

Before your first K8s Production Deployment



- Run K8s locally, as in your laptop!
- Get it up and running, deploy a sample container
- Configure  kubectl

I don't remember when I ran K8s first, or how many times I have run it locally or in various example labs on the internet. Recently I ran some K8s BGP labs in my browser

Trying out K8s on your laptop:

- Smallest demo with single node minikube running inside Debian 11 "start and run a hello world"
- Next one might be multiple VMs running K8s nodes – multi node with OpenBSD VM for BGP
- I can run all of this inside Qubes as HVM

Kubernetes: Production-Grade Container Orchestration



Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

It groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.

Source: <https://kubernetes.io/>

Basics:

- Create Cluster, Deploy App, Expose your app, Scale up, Update your app
- Plus a thousand buzz words – scaling, micro-services,

Don't get me wrong, I really like Kubernetes!

K8s Run Programs, Processes and Applications



Kubernetes is a platform for creating, deploying, and managing distributed applications. These applications come in many different shapes and sizes, but ultimately, they are all comprised of one or more programs that run on individual machines.

Source: *Kubernetes: Up and Running*, 3rd Edition, Brendan Burns, Joe Beda, Kelsey Hightower

- Containers – running processes and container images
- Interfaces API – management interfaces are of special interest, including things like Service accounts
- Ports and Services – running applications, what is exposed
- **K8s security is software security**

Initial security advice, read the documentation



Learn Kubernetes Basics

- Apply Pod Security Standards at the Cluster Level
- Apply Pod Security Standards at the Namespace Level
- Restrict a Container's Access to Resources with AppArmor
- Restrict a Container's Syscalls with seccomp

Source: tutorial linked from front page of <https://kubernetes.io/>,
<https://kubernetes.io/docs/tutorials/kubernetes-basics/>

- I like that security is listed within the first tutorial!
- This links to further documentation
- Security is also being added continuously

Pod Security Standards (PSA)



Pod Security Standards

The Pod Security Standards define three different *policies* to broadly cover the security spectrum. These policies are *cumulative* and range from highly-permissive to highly-restrictive. This guide outlines the requirements of each policy.

Profile	Description
Privileged	Unrestricted policy, providing the widest possible level of permissions. This policy allows for known privilege escalations.
Baseline	Minimally restrictive policy which prevents known privilege escalations. Allows the default (minimally specified) Pod configuration.
Restricted	Heavily restricted policy, following current Pod hardening best practices.

Source: <https://kubernetes.io/docs/concepts/security/pod-security-standards/>

- Note: versions from 1.23, with varying features added later
"Pod Security admission (PSA) is enabled by default in v1.23 and later, as it graduated to beta."

Restricting the Kubernetes API



By default, the Kubernetes API server listens on port 6443 on the first non-localhost network interface, protected by TLS. In a typical production Kubernetes cluster, the API serves on port 443. The port can be changed with the `--secure-port`, and the listening IP address with the `--bind-address` flag.

...

If your cluster uses a **private certificate authority, you need a copy of that CA certificate configured into your `/.kube/config` on the client**, so that you can trust the connection and be confident it was not intercepted.

- If an attacker can run Kubectl with credentials – they can do anything
- Defense in Depth is suggested – so restrict at multiple levels
- Only allow connections from specific network – firewall filtering
- Use TLS and verify/copy certificates

Security From low level and outside



Now you have built a cluster – great, is it secure? Does it even work, can you run it?

Keeping it short on purpose:

- Hardware - DRAC, ILO, KVM, IPMI etc.
- Connections - how to connect your clusters
- Have some jump host, perhaps OpenBSD with wireguard VPN

Consider threats and access from the outside and inwards, from bottom and upwards

Document or it didn't happen



The hypothesis:

Amid a wash of paper, a small number of documents become the critical pivots around which every project's management revolves. These are the manager's chief personal tools.

- Make sure to document!
- There are so many parts, so many decisions, sooo many places thing can mess up
- Have a minimum of documentation – and the YAML files are not enough
- Hint: you can read the Mythical Man Month at <https://archive.org/details/MythicalManMonth>

Role-based Access Control (RBAC)



One thing you should document, create a policy for is access and permissions.

In computer systems security, **role-based access control (RBAC)**[1][2] or role-based security[3] is an approach to restricting system access to unauthorized users. It is used by the majority of enterprises with more than 500 employees,[4] and can implement mandatory access control (MAC) or discretionary access control (DAC).

Role-based access control (RBAC) is a policy-neutral access-control mechanism defined around **roles and privileges**. The components of RBAC such as role-permissions, user-role and role-role relationships make it simple to perform user assignments. A study by NIST has demonstrated that RBAC addresses many needs of commercial and government organizations[citation needed]. RBAC can be used to facilitate administration of security in large organizations with hundreds of users and thousands of permissions. Although RBAC is different from MAC and DAC access control frameworks, it can enforce these policies without any complication.

Quote from https://en.wikipedia.org/wiki/Role-based_access_control

Kubernetes Initial RBAC



The `admin.conf` file gives the user superuser privileges over the cluster. This file should be used sparingly. For normal users, it's recommended to generate an unique credential to which you grant privileges. You can do this with the `kubeadm alpha kubeconfig user --client-name <CN>` command. That command will print out a KubeConfig file to STDOUT which you should save to a file and distribute to your user. After that, grant privileges by using `kubectl create (cluster)rolebinding`.

- Right after install you can run 🔑 `kubectl - great`
- Age old advice do not use shared administration accounts – have individual accounts

Adding administrators



```
kubeadm config print init-defaults > kubeadm.conf  
kubeadm kubeconfig user --client-name=hlkadmin --config=kubeadm.conf > hlkadmin.conf
```

- I would recommend having multiple configs, even for yourself
- hlkadmin config used for administration
- hlkdeploy for deploying
- hlktester maybe even a testing user
- Read more about <https://kubernetes.io/docs/reference/access-authn-authz/rbac/>

RBAC, groups and Namespace



Kubernetes starts with three initial namespaces:

- * default The default namespace for objects with no other namespace
- * kube-system The namespace for objects created by the Kubernetes system
- * kube-public This namespace is created automatically and is readable by all users (including those not authenticated).

Source: <https://kubernetes.io/docs/tasks/administer-cluster/namespaces/>

- I like namespaces, easy way to isolate stuff
- Allow for quotas – don't allow a test application to use all resources
- Easy network segregation – network policies
- Allows easy data isolation – who can access sensitive data
- I recommend groups, namespaces and roles for granting access – **do not grant permission to named users!**

Installing tools and plugins



```
hlk@k8s-1:~$ kubectl krew update
Updated the local copy of plugin index.
hlk@k8s-1:~$ kubectl krew install rbac-view
Updated the local copy of plugin index.
Installing plugin: rbac-view
Installed plugin: rbac-view
\
| Use this plugin:
|   kubectl rbac-view
| Documentation:
|   https://github.com/jasonrichardsmith/rbac-view
| Caveats:
| \
|   | Run "kubectl rbac-view" to open a browser with an html view of your permissions.
| /
/
/
```

- "This project is considered prerelease and is under active development."
- What I have found in many books – tools, hints and tips, above from:
Hacking Kubernetes: Threat-Driven Analysis and Defense, Andrew Martin, Michael Hausenblas

Example tool RBAC-view



Projects frontend - Chromium

frontend 127.0.0.1:8800/index.html

RBAC

Cluster Roles Roles

cluster Search Fields..

ClusterRoles

RoleName	*	*scale	bindings	certificatesigningrequests	certificatesigningrequests/approval	certificatesigningrequests/indeedclient	certificatesigningrequests/selfindeedclient	certificatesigningrequests/status	ciliumbgppeeringservices	ciliumclusterwideenvoyconfigs	ciliumclusterwidennetworkpolicies	ciliumendpoints/status	ciliumendpoints	ciliumendpoints/status	ciliumendpoints/locations	ciliumendpoints/policies	ciliumendpoints/status	ciliumendpoints/trafficpools	ciliumloadbalancerip pools	ciliumloadbalancerip pools/status	ciliumlocalredirectpolicies	ciliumnetworkpolicies	ciliumnetworkpolicies/status	ciliumnodeconfigs	ciliumnodes	ciliumnodes/status	clusterroles	configmaps	controllerrevi		
system:controller:clusterrole-aggregation-controller																															
Subjects																															
cluster-admin																															
Subjects																															

<https://github.com/jasonrichardsmith/rbac-view>

RBAC Audit tools



Multiple tools exist – I most often only link open source tools in my presentations, feel free to suggest others

- 🔐 rbac-view – view RBAC data interactively and search as shown above <https://github.com/jasonrichardsmith/rbac-view> via:
Hacking Kubernetes: Threat-Driven Analysis and Defense, Andrew Martin, Michael Hausenblas
- 🔐 rback – generates a graph representation (in Graphviz dot format) of a Kubernetes cluster's RBAC settings <https://github.com/team-soteria/rback>
- 🔐 rbac-tool – A collection of Kubernetes RBAC tools to sugar coat Kubernetes RBAC complexity <https://github.com/alcideio/rbac-tool> via Rapid7
- 🔐 KubiScan – a tool by Eviatar Gerzi to scan Kubernetes cluster for risky RBAC permissions <https://github.com/cyberark/KubiScan>
- 🔐 krane – a Kubernetes RBAC static analysis and visualisation tool <https://github.com/appvia/krane>
- What is best current practice? More tools and articles available at: <https://rbac.dev/>

You MUST control your permissions, so find the tools that work for you!

Goals: Data Security



Nine principles of data security

- (1) **Access control**—Each identifiable clinical record shall be marked with an access control list naming the people or groups of people who may read it and append data to it. The system shall prevent anyone not on the list from accessing the record in any way.
- (2) **Record opening**—A clinician may open a record with herself and the patient on the access control list. When a patient has been referred she may open a record with herself, the patient, and the referring clinician(s) on the access control list.
- (3) **Control**—One of the clinicians on the access control list must be marked as being responsible. Only she may change the access control list and she may add only other health care professionals to it.
- (4) **Consent and notification**—The responsible clinician must notify the patient of the names on his record's access control list when it is opened, of all subsequent additions, and whenever responsibility is transferred. His consent must also be obtained, except in emergency or in the case of statutory exemptions.
- (5) **Persistence**—No one shall have the ability to delete clinical information until the appropriate time has expired.
- (6) **Attribution**—All accesses to clinical records shall be marked on the record with the name of the person accessing the record as well as the date and time. An audit trail must be kept of all deletions.
- (7) **Information flow**—Information derived from record A may be appended to record B if and only if B's access control list is contained in A's.
- (8) **Aggregation control**—Effective measures should exist to prevent the aggregation of personal health information. In particular, patients must receive special notification if any person whom it is proposed to add to their access control list already has access to personal health information on a large number of people.
- (9) **Trusted computing base**—Computer systems that handle personal health information shall have a subsystem that enforces the above principles in an effective way. Its effectiveness shall be evaluated by independent experts.

Source: *Clinical system security: Interim guidelines*, Ross Anderson, 1996

Principle of Least Privilege



Every program and every privileged user of the system should operate using the least amount of privilege necessary to complete the job. — Jerome Saltzer, Communications of the ACM

Also drop privileges when not needed anymore, relinquish rights immediately

Example, need to read a document - but not write.

Database systems can often provide very fine grained access to data

Read more in the 1975 article by Saltzer and Schroeder

https://en.wikipedia.org/wiki/Principle_of_least_privilege

https://en.wikipedia.org/wiki/Saltzer_and_Schroeder%27s_design_principles



Principle of Fail-Safe defaults

Definition 14-3 The *principle of fail-safe defaults* states that, unless a subject is given explicit access to an object, it should be denied access to that object.

Definition from *Computer Security*, Matt Bishop

Default access *none*

In firewalls default-deny - that which is not allowed is prohibited

Newer devices today can come with no administrative users, while older devices often came with default admin/admin users

Real world example, OpenSSH config files that come with PermitRootLogin no



Soft Multi tenancy

Depending on your needs, it might be interesting to investigate soft multi tenancy;

The level of isolation offered is sometimes described using terms like “hard” multi-tenancy, which implies strong isolation, and “soft” multi-tenancy, which implies weaker isolation. In particular, “hard” multi-tenancy is often used to describe cases where the tenants do not trust each other, often from security and resource sharing perspectives (e.g. guarding against attacks such as data exfiltration or DoS). Since data planes typically have much larger attack surfaces, “hard” multi-tenancy often requires extra attention to isolating the data-plane, though control plane isolation also remains critical.

Source: <https://kubernetes.io/docs/concepts/security/multi-tenancy/>

Goal prevent some avoidable accidents and problems, breaking other groups services, but easier to maintain

Network parts



Lets move to networking and CNI plugin alternatives

- Container Network Interface (CNI) and Cloud Native Computing Foundation (CNCF) project
<https://github.com/containernetworking/cni>
- Multiple options – popular and common ones Calico, Flannel, Weave and Cilium
- Calico - available multiple places
- What does changing CNI plugin bring? and importantly what do you need!
- Usually some tunneling and encapsulation between nodes, VXLAN, IP in IP
- Allow dynamic routing like Border Gateway Protocol (BGP)
- Often more advanced features for filtering, observability, ...
- Cilium - I have followed Cilium for a while, so I choose that

Focus on your needs, and not what their marketing materials say

Compare CNI Features



CNI	ENCRYPTION	NETWORK POLICIES
Calico		No Ingress + Egress
Canal		No Ingress + Egress
Cilium		Yes Ingress + Egress
Flannel		No
Kube-router		No Ingress only
WeaveNet		Yes Ingress + Egress

Source: *Learn Kubernetes Security* (LKS), Kaizhe Huang , Pranjal Jumde

- Can you live with kube-router really, probably not

Compare CNI Performance



CNI Benchmark August 2020 infraBuilder	Config	Performances (bandwidth)					Resources consumption (cpu/ram)					Security features				
		MTU		Pod to Pod		Pod to Service		Idle		Pod to Pod		Pod to Service		Network Policies		Encryption
	setting	TCP	UDP	TCP	UDP	none	TCP	UDP	TCP	UDP	TCP	UDP	in	out	activation	Performance
Antrea	auto	Very fast	Very fast	Very fast	Slow	Low	Low	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow
Calico	manual	Very fast	Very fast	Very fast	Fast	Low	Very low	Very low	Very low	Very low	Very low	Very low	yes	yes	anytime	Very fast
Canal	manual	Very fast	Very fast	Very fast	Very fast	Low	Very low	Very low	Very low	Very low	Very low	Very low	yes	yes	no	n/a
Cilium	auto	Fast	Very fast	Very fast	Very fast	High	High	High	High	High	High	High	yes	yes	at deploy time	Slow
Flannel	auto	Very fast	Very fast	Very fast	Very fast	Very low	Very low	Very low	Very low	Very low	Very low	Very low	no	no	no	n/a
Kube-OVN	auto	Fast	Very slow	Fast	Very slow	High	High	High	High	High	High	High	yes	yes	no	n/a
Kube-router	none	Slow	Very slow	Slow	Very slow	Low	Very low	Low	Very low	Low	Very low	Low	yes	yes	no	n/a
Weave Net	manual	Very fast	Very fast	Very fast	Fast	Very low	Low	Low	Low	Low	Low	Low	yes	yes	at deploy time	Slow

Source: <https://platform9.com/blog/the-ultimate-guide-to-using-calico-flannel-weave-and-cilium/>

- What is most important for you, features or performance? I cannot tell you which CNI to choose!

Installing Cilium



```
helm repo add cilium https://helm.cilium.io/  
  
API_SERVER_IP=<your_api_server_ip>  
# Kubeadm default is 6443  
API_SERVER_PORT=<your_api_server_port>  
helm install cilium cilium/cilium --version 1.12.5 \  
  --namespace kube-system \  
  --set kubeProxyReplacement=strict \  
  --set k8sServiceHost=${API_SERVER_IP} \  
  --set k8sServicePort=${API_SERVER_PORT}
```

- Note: I ended up deciding to run without the kube-proxy, only Cilium
- Document!



Document what you did, what it did

```
cilium install --version=1.13.0-rc4 \
    --helm-set ipam.mode=kubernetes --helm-set tunnel=disabled \
    --helm-set ipv4NativeRoutingCIDR="10.0.0.0/8" --helm-set bgpControlPlane.enabled=true \
    --helm-set k8s.requireIPv4PodCIDR=true --helm-set kube-proxy-replacement=strict

Using Cilium version 1.13.0-rc4
Auto-detected cluster name: kubernetes
Auto-detected datapath mode: tunnel
Auto-detected kube-proxy has not been installed
Cilium will fully replace all functionalities of kube-proxy
helm template --namespace kube-system cilium cilium/cilium --version 1.13.0-rc4 --set bgpControlPlane.enabled=true,
cluster.id=0,cluster.name=kubernetes,encryption.nodeEncryption=false,ipam.mode=kubernetes,
ipv4NativeRoutingCIDR=10.0.0.0/8,k8s.requireIPv4PodCIDR=true,k8sServiceHost=10.137.0.26,
k8sServicePort=6443,kube-proxyreplacement=strict,kubeProxyReplacement=strict,operator.replicas=1,
serviceAccounts.cilium.name=cilium,serviceAccounts.operator.name=cilium-operator,tunnel=disabled
...
Waiting for Cilium to be installed and ready...
Cilium was successfully installed! Run 'cilium status' to view installation health
```

How I ran the kubeadm for building cluster - note the skip:

```
sudo kubeadm init --pod-network-cidr=10.50.0.0/16 --skip-phases=addon/kube-proxy
```

Cilium CLI tool

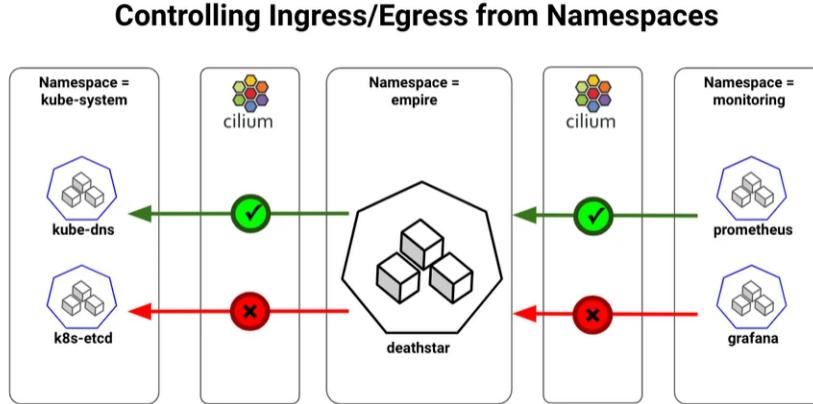


```
[Projects] Terminal - hlk@k8s-1: ~
File Edit View Terminal Tabs Help
root@k8s-1:~# cilium status
      /--\
      /_ \_/_\_
      \_/_\_\_/
      /_ \_/_\_\_
      \_/_\_\_/_\_
      Cilium:      OK
      Operator:    OK
      Hubble:     disabled
      ClusterMesh: disabled

Deployment      cilium-operator  Desired: 1, Ready: 1/1, Available: 1/1
DaemonSet       cilium          Desired: 2, Ready: 2/2, Available: 2/2
Containers:     cilium          Running: 2
                  cilium-operator Running: 1
Cluster Pods:   8/8 managed by Cilium
Image versions  cilium          quay.io/cilium/cilium:v1.13.0-rc4@sha256:32acd47fd9bea9c0045222ba5d27f5fe9ad06dabd572a80b870b1f0e68c0e928: 2
                cilium-operator   quay.io/cilium/operator-generic:v1.13.0-rc4@sha256:19f612d4f1052e26edf33e26f60d64d8fb6caed9f03692b85b429a4ef5d175b2: 1
root@k8s-1:~#
```

- Many tools are executed via kubectl
- Others have their own command
- This can be very confusing, and again – document which tools you use!
- Having a jump host with updated tools installed might help – helps me!

Cilium overview



Kubernetes provides Network Policies for controlling traffic going in and out of the pods. Cilium implements the Kubernetes Network Policies for L3/L4 level and extends with L7 policies for granular API-level security for common protocols such as HTTP, Kafka, gRPC, etc

Source: picture and text from <https://cilium.io/blog/2018/09/19/kubernetes-network-policies/>

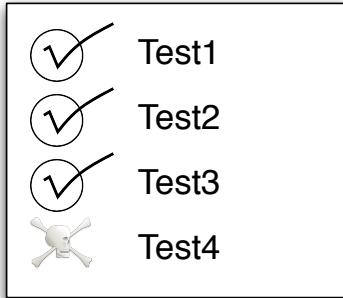


Testing Connectivity

Cilium has a very nice built-in connectivity test:

```
root@k8s-1:~# cilium connectivity test
  Monitor aggregation detected, will skip some flow validation steps
[kubernetes] Creating namespace cilium-test for connectivity check...
[kubernetes] Deploying echo-same-node service...
[kubernetes] Deploying DNS test server configmap...
[kubernetes] Deploying same-node deployment...
[kubernetes] Deploying client deployment...
[kubernetes] Deploying client2 deployment...
[kubernetes] Deploying echo-other-node service...
[kubernetes] Deploying other-node deployment...
[kubernetes] Waiting for deployments [client client2 echo-same-node] to become ready...
[kubernetes] Waiting for deployments [echo-other-node] to become ready...
...
All 31 tests (232 actions) successful, 0 tests skipped, 0 scenarios skipped.
root@k8s-1:~#
```

Checking your Kubernetes deployment



- It can actually be hard to check your Kubernetes installation
- Especially networking in detail
- I would recommend spending a little time investigating Maximum Transmission Unit (MTU)
https://en.wikipedia.org/wiki/Maximum_transmission_unit
- Your devices – switches and network cards can probably also offload some features to hardware
- Advanced users could dive deeper into *BPF and XDP Reference Guide* <https://docs.cilium.io/en/latest/bpf/>

External IPs



Jumping directly in, I am running on bare metal. Getting external access into K8s is a bit hard. I decided to use Cilium, BGP, and IP pools

```
$ cat pool-zencurity.yaml
---
apiVersion: "cilium.io/v2alpha1"
kind: CiliumLoadBalancerIPPool
metadata:
  name: "pool"
spec:
  cidrs:
    - cidr: "185.129.62.144/28"
      - cidr: "2a06:d380:0:85::0/64"
```

- <https://docs.cilium.io/en/latest/network/lb-ipam/>
- I run a LIR (Local Internet Registry) with network AS57860 which have these addresses



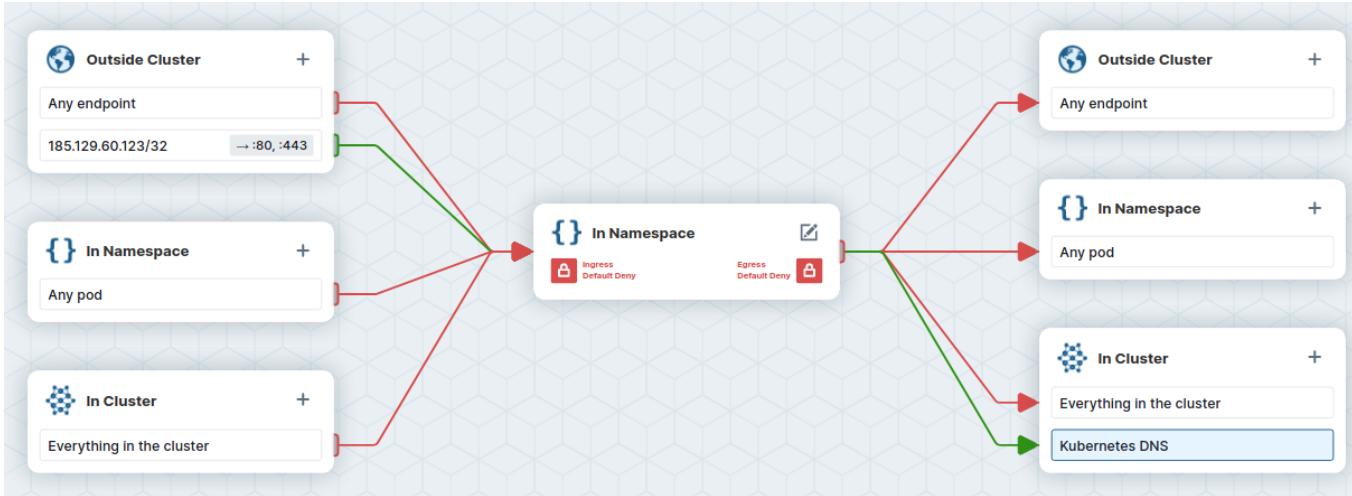
Result in my setup

```
root@k8s-1:/home/hlk/bin# k get svc
NAME           TYPE      CLUSTER-IP    EXTERNAL-IP        PORT(S)        AGE
kubernetes     ClusterIP   10.96.0.1    <none>          443/TCP       5d
service-test   LoadBalancer 10.108.127.95 185.129.62.154 1234:30378/TCP 4d22h
```

References that were useful:

- <https://sue.eu/blogs/expose-loadbalanced-kubernetes-services-with-bgp-cilium/>
- <https://nicovibert.com/2022/07/21/bgp-with-cilium/>
- <https://isovalent.com/resource-library/labs/> - has labs where you can try running BGP

Network Policy Editor for Kubernetes



- Nice Editor for Policies <https://editor.cilium.io/>
- Recommendation: always specify a Network Policy - be specific, who CAN access
- Make positive lists - as normally only a limited number of other resources will need access
- Then also make sure to specify an egress policy - so both ingress and egress rules

Example Network Policy



```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy      see more at https://docs.cilium.io/en/v1.8/concepts/kubernetes/policy/
metadata:
  name: example-policy
spec:
  podSelector:
  policyTypes:
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 185.129.60.123/32
      ports:
        - port: 80
        - port: 443
  egress:
    - to:
        - namespaceSelector:
          podSelector:
            matchLabels:
              k8s-app: kube-dns
  ports:
    - port: 53           protocol: UDP
```

Cilium for Host Firewall



Deploy Cilium release via Helm:

```
helm install cilium ./cilium --namespace kube-system --set hostFirewall.enabled=true --set devices='ethX,ethY'
```

The devices flag refers to the network devices Cilium is configured on such as eth0. Omitting this option leads Cilium to auto-detect what interfaces the host firewall applies to.

At this point, the Cilium-managed nodes are ready to enforce network policies.

Source: <https://docs.cilium.io/en/latest/security/host-firewall/>

- I find it cleaner to use Cilium for all network policies, ignoring IPTables
- You need only a few CiliumClusterwideNetworkPolicy resources defined for the cluster to work, and then you can allow Secure Shell from only a few places
- Note: While booting there might be a very short time where Cilium is starting, where packets are allowed YMMV

Monitoring and updating – upgrades



Since Kubernetes is a complex system, I recommend making an initial deployment plan which is longer than a release cycle of Kubernetes. Currently new releases are coming out rapidly, so a 3-4 month plan should suffice.

Main goal is to try upgrading the cluster *before* you have production running!

<https://kubernetes.io/releases/>

You need to upgrade all parts, so remember which parts you use:

- Firmwares and supporting systems, routers, switches, server control, storage, ...
- Operating systems
- Kubernetes core components
- Cilium upgrade and various other plugins

Performing upgrades



When rolling out an upgrade with Kubernetes, Kubernetes will first terminate the pod followed by pulling the new image version and then finally spin up the new image. In order to reduce the downtime of the agent and to prevent ErrImagePull errors during upgrade, the pre-flight check pre-pulls the new image version.

```
root@gouda01:~# cilium upgrade
Auto-detected datapath mode: tunnel
Upgrading cilium-operator to version quay.io/cilium/operator-generic:v1.12.5...
Upgrading cilium to version quay.io/cilium/cilium:v1.12.5...
Waiting for Cilium to be upgraded...
```

- Cilium recommend a pre-flight check – pulls images beforehand etc:
<https://docs.cilium.io/en/v1.12/operations/upgrade/>
- A good example of how to prepare upgrades

Resource monitoring and capacity planning



Metrics A series of numbers measured over a period of time

Logs Used for exploratory analysis of a system

Source: *Kubernetes Best Practices: Blueprints for Building Successful Applications on Kubernetes*, Brendan Burns, Eddie Villalba, Dave Strelak and Lachlan Evenson, O'Reilly 2020

- The book is mostly interested in poorly performing applications, but the same data is used in security
- I always install Kubernetes dashboard, and I am going to use 🔐 Prometheus a lot more



Security monitoring and auditing

Why isn't K8s secure?!

We will cover the following topics in this chapter:

- The path traversal issue in kubectl cp—CVE-2019-11246
- The DoS issue in JSON parsing—CVE-2019-1002100
- The DoS issue in YAML parsing—CVE-2019-11253
- The privilege-escalation issue in role parsing—CVE-2019-11247
- Scanning known vulnerabilities using 🔑 kube-hunter

Source: Chapter 13 *Learn Kubernetes Security* (LKS), Kaizhe Huang , Pranjal Jumde, Packt, July 2020

TL;DR look at previous problems – which ones would have hurt our design and architecture?

Cloud and Container Forensics



I am definitely not an expert here!

Tools like kube-forensics “create checkpoint snapshots of the state of running pods for later off-line analysis,” so malicious workloads can be dumped and killed, and the system returned to use.

Source: *Hacking Kubernetes: Threat-Driven Analysis and Defense*, Andrew Martin, Michael Hausenblas, O'Reilly, October 2021

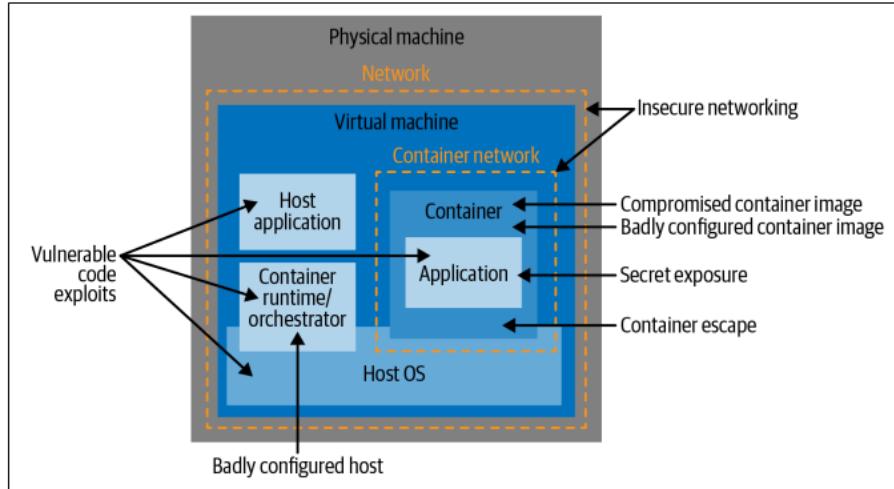
- Sysdig - advanced logging and inspection
- 🔐 sysdig and 🔐 sysdig-inspect <https://github.com/draios/sysdig> <https://github.com/draios/sysdig-inspect>

Monitoring Networking



- Cilium has Setting up Hubble Observability
https://docs.cilium.io/en/stable/gettingstarted/hubble_setup/#hubble-setup
- Attack types DDoS slow and fast, volumetric and PPS based
- Pro/cons with having another layer - manually configured, only allow few protocols and ports 80, 443, 8080
- If you have allow lists only, you will reduce noise

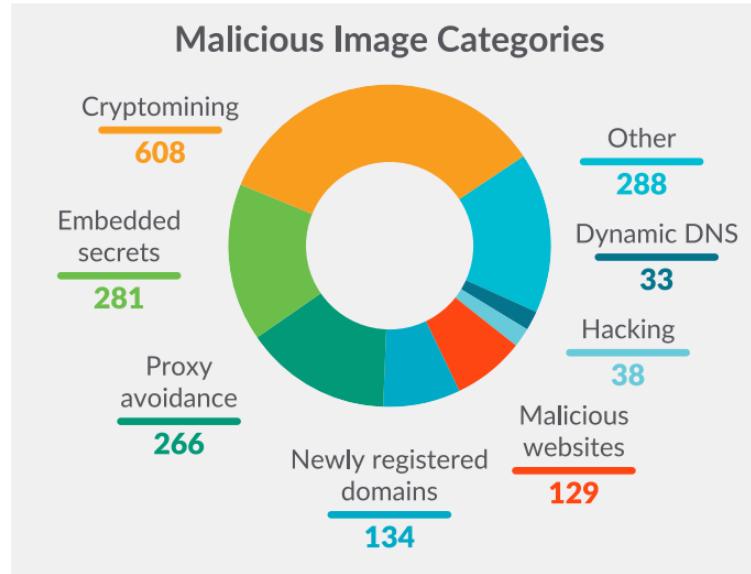
Kubernetes Security is more than just the core



Source: *Container Security*, (CS) Liz Rice, O'Reilly, Apr 2020

- Attacks from inside, attacks inside containers
- Some just use up resources, some attacks others, some attack the system from the inside
- Hacking Kubernetes have a whole *Chapter 3 Container Runtime Isolation*

Analysis on Docker Hub malicious images: Attacks through public container images



This article is relevant, talking about malicious docker images

<https://sysdig.com/blog/analysis-of-supply-chain-attacks-through-public-docker-images/>

Keeping Container Images secure



- 🔑 anchore open-source project that provides a centralized service for inspection, analysis, and certification of container images <https://github.com/anchore/anchore-engine>
"As of 2023, Anchore Engine is no longer maintained. There will be no future versions released. Users are advised to use Syft and Grype."
- 🔑 Syft <https://github.com/anchore/syft> and 🔑 Grype <https://github.com/anchore/grype>
- Allow direct download from the internet into your cluster, may become a problem
- Malicious people are typosquatting popular containers!
- Supply chain attacks in general are a problem

Harden Container Images and Update Your Procedures



- Change the goddamn passwords!
Container postgresql with user postgres and password *postgres*, REALLY!!!!!!1111
- and NO MORE ROOT! Dont run as root, we realized this was bad in the 1990s!
- CIS Docker Benchmarking also Learn Kubernetes Security *Chapter 8: Securing Kubernetes Pods*
- Hacking Kubernetes *Chapter 8: Policy* - describe things like Resource Quotas, Runtime Policies



Benchmarking tools

💡 Kube-bench is the industry-standard tool to automate checking Kubernetes compliance with the Center for Internet Security (CIS) Benchmark.

Kube-bench makes it easy for operators to check whether each node in their Kubernetes cluster is configured according to security best practices.

Source: <https://info.aquasec.com/open-source>

- CIS Kubernetes V1.24 Benchmark v1.0.0 - 09-21-2022 – other versions exist
- CIS Docker Benchmark v1.5.0 - 12-28-2022

See more at <https://www.aquasec.com/cloud-native-academy/docker-container/docker-cis-benchmark/>



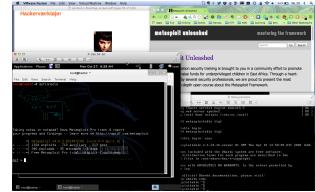
Tool example kube-bench

```
hlk@timon:~/bin/kube-bench/kube-bench$ kubectl logs kube-bench-gdf62
[PASS] 1.1.7 Ensure that the etcd pod specification file permissions are set to 600 or more restrictive (Automated)
[PASS] 1.1.8 Ensure that the etcd pod specification file ownership is set to root:root (Automated)
[WARN] 1.1.9 Ensure that the Container Network Interface file permissions are set to 600 or more restrictive (Manual)
[WARN] 1.1.10 Ensure that the Container Network Interface file ownership is set to root:root (Manual)
[PASS] 1.1.11 Ensure that the etcd data directory permissions are set to 700 or more restrictive (Automated)
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)
...
== Summary policies ==
0 checks PASS
0 checks FAIL
35 checks WARN
0 checks INFO

== Summary total ==
63 checks PASS
10 checks FAIL
58 checks WARN
0 checks INFO
```

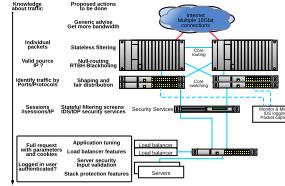
- <https://github.com/aquasecurity/kube-bench> also check out Lynis <https://cisofy.com/lynis/>

Kubernetes lab setup – proof of concept



- Hardware: any modern PC with modern CPU and virtualisation
Don't forget to enable it in the BIOS
- Software: your favourite Linux distribution, I choose Debian
- Container software: pick your poison
- Hacker software: Kali as a Virtual Machine <https://www.kali.org/>
- Smallest Kubernetes: Minikube - I run this on Debian 11
- Production deployment probably would use better tools like 🔧 Kubeadm

Conclusion Kubernetes Security



- It is hard!
- Multiple books have checklist – depending on your interests, some for developers, some for ops
- Example: Hacking Kubernetes chapter 10. has good advice too, including a small checklist for On-Premises Environments links on p249 to Build K8s Bare-metal cluster with external access
<https://medium.com/swlh/on-premise-kubernetes-clusters-b36660ca6914>
<https://www.datapacket.com/blog/build-kubernetes-cluster>
<https://medium.com/@apiotrowski312/bare-metal-kubernetes-with-helm-rook-ingress-prometheus-grafana-6a74857cc74>

Questions?



Henrik Kramselund he/him han/ham hlk@zencurity.com @kramse

You are always welcome to send me questions later via email

Mobile: +45 2026 6000

Email: hlk@zencurity.com

Let's talk about a case Cheesehub.dk



CHEESE: Cyber Human Ecosystem of Engaged Security Education

CHEESE (Cyber Human Ecosystem of Engaged Security Education) is an NSF-funded initiative to develop a learning ecosystem for cybersecurity education. The project consists of the following:

- Easy to access, dynamic, web-based learning platform
 - Community-contributed catalog of cybersecurity training materials
 - Community-driven platform to request, develop, share, evaluate and learn about content
 - Learning ecosystem that is continuously updated
-
- I found a system based on Kubernetes
CHEESE: Cyber Human Ecosystem of Engaged Security Education
<https://docs.cheesehub.org/>
 - Educational sessions using containers being spun up for the students
 - Decided to give it a shot

Hacker Days and NWWC



Due to time restrictions and my knowledge, some things have moved to Advanced Subjects

We can also consider doing focused work days with Kubernetes – I would love that!

I have previously done NWWC see nwwc.dk, Hacker days at PROSA, KEA etc.

Do you want to meet up?

Threat modelling – maybe do your own threat modelling?



- my presentation based on "minimum" and Best Current Practice

Make sure to do recommended basic security before considering advanced attacks

Advanced subjects



- Secrets in K8s vaults Various methods for keeping secrets – considered a developer problem, sorry
- High availability Moved to advanced subject, but of course important
- Multi tenancy Moved to advanced subject, perhaps HK chapter 7 can help you
- Sandboxing see HK Chapter 3. Sandboxing
- Honeypots Moved to advanced - time constraints



Books and educational materials

We could keep recommending books, articles, papers and official documentation. These are examples, and even if some details are changed, may be good to read

- *Kubernetes Security and Observability* Brendan Creane, Amit Gupta Mastering Kubernetes, **Third Edition**, Gigi Sayfan, Packt 2020
- *kubectl: Command-Line Kubernetes in a Nutshell*, Rimantas Mocevicius, Packt, 2020
- *The Kubernetes Workshop*, Zachary Arnold, Sahil Dua, Wei Huang, Faisal Masood, Melony Qin, and Mohammed Abu Taleb, Packt, 2020
- *Kubernetes A Complete DevOps Cookbook*, Murat Karslioglu, Packt, 2020
- *Cloud Native with Kubernetes*, Alexander Raul, Packt, 2020
- *Kubernetes and Docker – An Enterprise Guide*, Scott Surovich, Marc Boershtein, Packt, 2020
- *Kubernetes in Production Best Practices: Build and manage highly available production-ready Kubernetes clusters*, Aly Saleh, Murat Karslioglu, Packt 2021

These are the ones I selected for *my Kubernetes education* some via a Humble Ultimate Devops bundle

Linux and Internet Resources



Let's face it, K8s is Unix, so basic Linux and Internet Security resources help too:

- *The Linux Command Line: A Complete Introduction*, 2nd Edition
by William Shotts, internet edition <https://sourceforge.net/projects/linuxcommand>
- *Gray Hat Hacking: The Ethical Hacker's Handbook*, 5. ed. Allen Harper and others ISBN: 978-1-260-10841-5
- *Defensive Security Handbook: Best Practices for Securing Infrastructure*, Lee Brotherton, Amanda Berlin ISBN: 978-1-491-96038-7 284 pages
- *Web Application Security*, Andrew Hoffman, 2020, ISBN: 9781492053118
- *Practical Packet Analysis, Using Wireshark to Solve Real-World Network Problems* by Chris Sanders, 3rd ed, ISBN: 978-1-59327-802-1

We teach using these books and others! Diploma in IT-security at KEA Kompetence
<https://zencurity.gitbook.io/>