



Welcome to

Experiences with a small on-premise Kubernetes setup

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

Slides are available as PDF, kramse@Codeberg
kubernetes-on-premise.tex in the repo security-courses

Contact information



- Henrik Kramselund, he/him internet samurai mostly networks and infosec
- Network and security consultant Zencurity, teacher and activist
- Master from the Computer Science Department at the University of Copenhagen (DIKU)
- Email: hlk@zencurity.com Mobile: +45 2026 6000
- Run a small network for fun AS57860

You are welcome to drop me an email



Run K8s Anywhere

Kubernetes is open source giving you the freedom to take advantage of on-premises, hybrid, or public cloud infrastructure, letting you effortlessly move workloads to where it matters to you.

To download Kubernetes, visit the [download](#) section.

Disclaimer:

Kubernetes is everywhere and can support many interesting use-cases for hosting services, while some feel it is overkill - decide for yourself.

This talk is based on my experience as a Unix, Internet and Security person with researching K8s over some years and setting up my own infrastructure with Kubernetes in a small production setup.

Use the fine manual at the main site: <https://kubernetes.io/>

On-premise migration can also be part of digital sovereignty and data independence

Goals for today



- Experiences running the setup for one more year now
- How do you bring up a bare-metal test-cluster using Kubeadm, I use two small VMs for all of it
- Cilium as the network plugin - and listing some things I noticed, like your setup may be correct, but the version of cilium can have a problem
- Keeping your cluster connected to the outside world with BGP, worked nicely
- Show a Kubernetes lab and run some tools
- Point you towards resources, so you can get started with Kubernetes more securely. Plan for a modern featureful reasonably secure K8s bare-metal install – works for me

The garden



A screenshot of a Chromium browser window titled "[Projects] My digital garden garden.kramse.org - Chromium". The address bar shows "My digital garden garden.kramse.org". The page content includes a navigation bar with "My digital garden garden.kramse.org", "About", and "Library" links. Below that is a "Welcome!" message with a small green plant icon, followed by a paragraph of text about the digital garden.

I use this to run a small set of services based on Nginx as ingress controller, and Nginx serving applications (very boring static sites generated using Jekyll) you can see the result at: <https://garden.kramse.org> which also documents this setup

TL;DR My setup has now run for quite some time, and is very stable. Comparable to my OpenBSD web servers and also on par with public clouds. Runs inside the Adeo Datacenter <https://adeodc.dk/>

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.

Target Audience



- Overall target audience: operators, and developers becoming operators – devops
- Maybe secdevops - security personnel that needs some scalable systems running on K8s - like myself
- People being told to setup a K8s cluster – for some app

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, ...

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:

- I run two VMs for the K8s nodes – Debian 12
- Run NFS for storage – very easy to bind into the nodes
- Choose the Cilium for networking <https://cilium.io/>
- Cluster installed and maintained with Kubeadm <https://kubernetes.io/docs/reference/setup-tools/kubeadm/>

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>
- We also use decision records in my company:
<https://github.com/joelparkerhenderson/architecture-decision-record/>
<https://github.com/joelparkerhenderson/decision-record/>

Why Have Formal Documents



Why Have Formal Documents?

First, writing the decisions down is essential. Only when one writes do the gaps appear and the inconsistencies protrude. The act of writing turns out to require hundreds of mini-decisions, and it is the existence of these that distinguishes clear, exact policies from fuzzy ones.

Second, the documents will communicate the decisions to others. The manager will be continually amazed that policies he took for common knowledge are totally unknown by some member of his team. Since his fundamental job is to keep everybody going in the same direction, his chief daily task will be communication, not decision-making, and his documents will immensely lighten this load.

Finally, a manager's documents give him a data base and checklist. By reviewing them periodically he sees where he is, and he sees what changes of emphasis or shifts in direction are needed.

Source: *The Mythical Man-Month (Anniversary Edition)* by Frederick P. Brooks Jr.

- I use Git version control for my documentation purposes, some is in Zim Personal Wiki <https://zim-wiki.org/>
- I also use an IPAM <https://spritelink.github.io/NIPAP/>
- <https://garden.kramse.org> which also documents this setup

Adding administrators for people



```
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 namespaces



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).

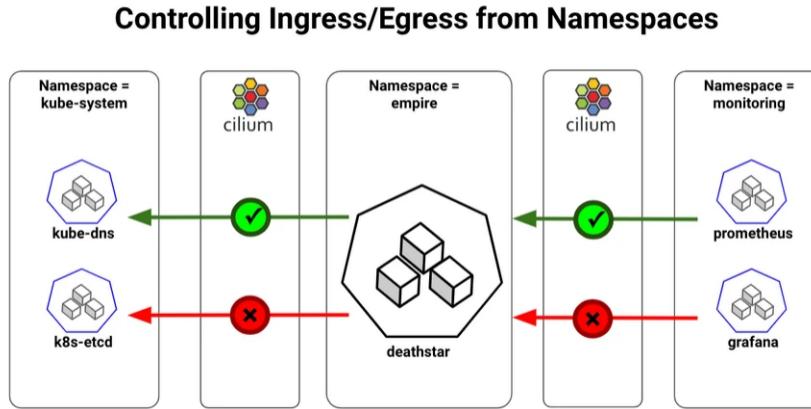
Source: <https://kubernetes.io/docs/tasks/administer-cluster/namespaces/>

- 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!**

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
- Cilium - I have followed Cilium for a while, so I choose that

Installing Cilium - part 1



```
hlk@cheese01:~/bin$ cat install-cilium.sh
#!/bin/sh
# Find releases with: cilium install --list-versions
cilium uninstall
echo waiting 15 seconds
sleep 15

API_SERVER_IP=::1
API_SERVER_PORT=6443

# May 27, 2025 this version was installed
#VERSION=1.16.3
# New version
#VERSION=1.17.4
# Not working with IPv6! Sounds like: https://github.com/cilium/cilium/issues/37817
# Reverted to 1.16, but newer version .10
VERSION=1.16.10
```

- As can be read, there has been problems along the way
- Ended up doing this complete uninstall and re-install, only a few minutes on my setup

Installing Cilium - part 2



```
# Gateway API CRDs
# Read on https://isovalent.com/blog/post/tutorial-getting-started-with-the-cilium-gateway-api/ it needs to be installed FIRST
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.2.0/standard-install.yaml
kubectl apply -f https://github.com/kubernetes-sigs/gateway-api/releases/download/v1.2.0/experimental-install.yaml

# Kubernetes was installed using these:
# So make sure to set the same!
NATIVE_CIDR=10.50.0.0/16
NATIVE_CIDR6="2a06:d380:9984:0::/96"

cilium install --version=$VERSION --helm-set ipam.mode=cluster-pool
--helm-set bgpControlPlane.enabled=true --helm-set k8s.requireIPv4PodCIDR=true --helm-set kube-proxy-replacement=strict
--helm-set prometheus.enabled=true          --helm-set operator.prometheus.enabled=true
--helm-set hubble.relay.enabled=true        --helm-set hubble.ui.enabled=true
--helm-set hubble.metrics.enabled="{dns,drop,tcp,flow,icmp,http}" --helm-set
--policy-audit-mode=true --helm-set hostFirewall.enabled=true --helm-set devices='{enp1s0}'
--helm-set namespace=kube-system --helm-set ipv6.enabled=true --set enableIPv6Masquerade=false
--helm-set ipam.operator.clusterPoolIPv4PodCIDRList=$NATIVE_CIDR
--helm-set ipam.operator.clusterPoolIPv6PodCIDRList=$NATIVE_CIDR6
--helm-set ipam.operator.clusterPoolIPv4MaskSize=24 --helm-set ipam.operator.clusterPoolIPv6MaskSize=112
--helm-set bpf.masquerade=true --helm-set gatewayAPI.enabled=true
```



Installing Cilium - Finish up

We are missing the following lines from the install script, restart the thing:

```
#kubectl rollout restart daemonset -n kube-system cilium  
kubectl -n kube-system rollout restart deployment/cilium-operator
```

Overall experience with cilium - great!



- I have tried both release candidates, production releases and had few problems
- It is quite easy to install cilium with the script, try a new / newer version
- It starts up quickly, and fails equally fast – where you can read why it didn't start
- My initial rules for Cilium has been very stable - including as the host firewall too!
<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

Worst when learning something new, trust yourself! You read the damn manual, but it didn't work - it might be a bug!

Document what you did, what it did



Initially I started out with much fewer options for cilium:

```
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
```

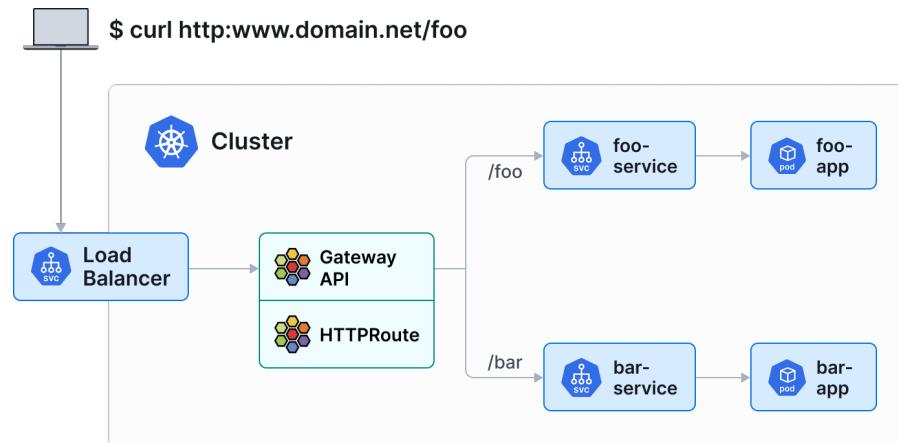
Cilium Gateway and TLS



Initially I had started with NGINX Ingress and Cilium, quite simple in the cloud world.

Along the way I found out there is a new thing called Gateway API

<https://kubernetes.io/docs/concepts/services-networking/gateway/>



Source: <https://cilium.io/use-cases/gateway-api/>

Gateway API in real life



```
---  
apiVersion: gateway.networking.k8s.io/v1beta1  
kind: Gateway  
metadata:  
  name: garden-gateway  
  namespace: webservices  
  annotations:  
    cert-manager.io/cluster-issuer: "letsencrypt-prod"  
spec:  
  gatewayClassName: cilium  
  infrastructure:  
    annotations:  
      io.cilium/lb-ipam-ips: 185.129.62.180, 2a06:d380:0:9985::1  
    # Not in 1.16.3 yet  
    # addresses:  
    #   - type: IPAddress  
    #     value: 185.129.62.177
```

- I run my DNS separate from K8s, so I allocate static IPs externally and add them here
- Also again note that some feature was not in the version I used originally etc.

Gateway API listeners



```
listeners:
  - name: http-garden
    protocol: HTTP
    port: 80
    hostname: garden.kramse.org
    allowedRoutes:
      namespaces:
        from: All
    ...
    more port 80 listeners
  - name: https-garden
    protocol: HTTPS
    port: 443
    hostname: garden.kramse.org
    tls:
      mode: Terminate
      certificateRefs:
        - kind: Secret
          name: garden-tls
    allowedRoutes:
      namespaces:
        from: All
```

Gateway API HTTPRoute with TLS



```
apiVersion: gateway.networking.k8s.io/v1beta1
kind: HTTPRoute
metadata:
  name: http-app-tls
  namespace: webservices
spec:
  parentRefs:
  - name: garden-gateway
    namespace: webservices
  hostnames:
  - "garden.kramse.org"
  rules:
  - matches:
    - path:
        type: PathPrefix
        value: /
  - backendRefs:
    - name: nginx-garden
      port: 80
```

- End result is very easy to follow, connects things

Cilium and other CLI tools

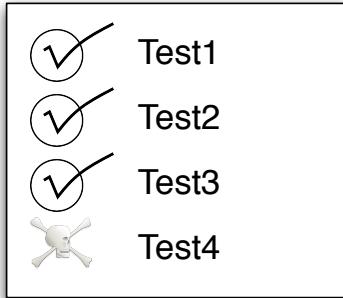


```
[Projects] Terminal - hlk@k8s-1: ~
File Edit View Terminal Tabs Help
root@k8s-1:~# cilium status
      /--\      Cilium:      OK
     /_ \_/_\_\_ Operator:      OK
    / \_/\_/\_/\_ Hubble:    disabled
   / \_/\_/\_/\_ ClusterMesh: disabled

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: kubeadm, crictl
- This can be very confusing, and again – document which tools you use!
- Having a **jump host with updated tools installed** might help – helps me!

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/>



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:~#
```

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

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
- This part has been running flawlessly, no more comments - nice!



Result in my setup

```
hlk@cheese01:~$ k get gateway -A
```

NAMESPACE	NAME	CLASS	ADDRESS	PROGRAMMED	AGE
webservices	garden-gateway	cilium	185.129.62.180	True	228d

```
hlk@cheese01:~$ k get svc -A
```

NAMESPACE	NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)
cert-manager	cert-manager	ClusterIP	10.96.2.250	<none>	9402/TCP
cert-manager	cert-manager-cainjector	ClusterIP	10.96.14.165	<none>	9402/TCP
cert-manager	cert-manager-webhook	ClusterIP	10.96.214.76	<none>	443/TCP,9402/TCP
default	kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP
kube-system	cilium-envoy	ClusterIP	None	<none>	9964/TCP
kube-system	kube-dns	ClusterIP	10.96.0.10	<none>	53/UDP,53/TCP,915
webservices	cilium-gateway-garden-gateway	LoadBalancer	10.96.85.230	185.129.62.180,2a06:d380:0:9985::1	80:30137/TCP,443:
webservices	nginx-files-kramse	LoadBalancer	10.96.234.56	185.129.62.176,2a06:d380:0:9985::2	80:30854/TCP,443:
webservices	nginx-garden	LoadBalancer	10.96.49.30	185.129.62.179,2a06:d380:0:9985::	80:31498/TCP,443:
webservices	nginx-library-kramse	LoadBalancer	10.96.44.145	185.129.62.177,2a06:d380:0:9985::3	80:30420/TCP,443:

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: Cilium



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



Performing upgrades: CRI-O and K8s

Basically I follow the instructions from:

<https://kubernetes.io/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/>

Example in mid May 2025, and 1.33 came out recently. I upgraded from 1.31 over 1.32 to 1.33, without problems

The process was for each upgrade

- Update repositories
- Upgrade kubeadm from say 1.32 which you are running to 1.33 which is the version you want
- Run kubeadm upgrade plan to check a few things
- Perform upgrade on controller to the version you want, like kubeadm upgrade apply v1.33.1
- Upgrade software, CRI-O container, kubectl and kubelet etc. just regular apt update/upgrade
- Do a reboot, controlled, chaotic, whatever :-D



kubeadm upgrade plan

```
# kubeadm upgrade plan
[preflight] Running pre-flight checks.
[upgrade/config] Reading configuration from the "kubeadm-config" ConfigMap in namespace "kube-system"...
[upgrade/config] Use 'kubeadm init phase upload-config --config your-config-file' to re-upload it.
W0525 20:44:37.362442 833945 configset.go:78] Warning: No kubeProxy.config.k8s.io/v1alpha1 config is loaded.
Continuing without it: configmaps "kube-proxy" not found
[upgrade] Running cluster health checks
[upgrade] Fetching available versions to upgrade to
[upgrade/versions] Cluster version: 1.32.5
[upgrade/versions] kubeadm version: v1.33.1
[upgrade/versions] Target version: v1.33.1
[upgrade/versions] Latest version in the v1.32 series: v1.32.5

W0525 20:44:44.068842 833945 configset.go:78] Warning: No kubeProxy.config.k8s.io/v1alpha1 config is loaded.
Continuing without it: configmaps "kube-proxy" not found
Components that must be upgraded manually after you have upgraded the control plane with 'kubeadm upgrade apply':
COMPONENT      NODE      CURRENT      TARGET
kubelet        cheese01    v1.32.5     v1.33.1
kubelet        cheese02    v1.32.5     v1.33.1
```

continued



Upgrade to the latest stable version:

COMPONENT	NODE	CURRENT	TARGET
kube-apiserver	cheese01	v1.32.5	v1.33.1
kube-controller-manager	cheese01	v1.32.5	v1.33.1
kube-scheduler	cheese01	v1.32.5	v1.33.1
kube-proxy		1.32.5	v1.33.1
CoreDNS		v1.11.3	v1.12.0
etcd	cheese01	3.5.16-0	3.5.21-0

You can now apply the upgrade by executing the following command:

```
kubeadm upgrade apply v1.33.1
```

Then after running the apply:

```
[upgrade] SUCCESS! A control plane node of your cluster was upgraded to "v1.33.1".
```

Logs from my upgrade can be seen at <https://garden.kramse.org/kubernetes-upgrade.html>

Package installation: Kubernetes and CRI-O repositories



Then following the instructions from the packaging README.md:

<https://github.com/cri-o/packaging/blob/main/README.md>

```
KUBERNETES_VERSION=v1.33
```

```
CRIO_VERSION=v1.33
```

```
curl -fsSL https://pkgs.k8s.io/core:/stable:/$KUBERNETES_VERSION/deb/Release.key |  
gpg --dearmor -o /etc/apt/keyrings/kubernetes-apt-keyring.gpg  
  
echo "deb [signed-by=/etc/apt/keyrings/kubernetes-apt-keyring.gpg]  
https://pkgs.k8s.io/core:/stable:/$KUBERNETES_VERSION/deb/ /" |  
tee /etc/apt/sources.list.d/kubernetes.list  
  
echo "and CRI-O"  
curl -fsSL https://download.opensuse.org/repositories/isv:/cri-o:/stable:/$CRIO_VERSION/deb/Release.key |  
gpg --dearmor -o /etc/apt/keyrings/cri-o-apt-keyring.gpg  
  
echo "deb [signed-by=/etc/apt/keyrings/cri-o-apt-keyring.gpg]  
https://download.opensuse.org/repositories/isv:/cri-o:/stable:/$CRIO_VERSION/deb/ /" |  
tee /etc/apt/sources.list.d/cri-o.list
```

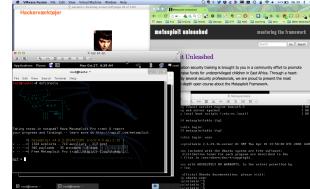
The software upgrade



```
# apt upgrade
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Calculating upgrade... Done
The following packages will be upgraded:
  cri-o cri-tools kubectl kubelet
4 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
Need to get 66.5 MB of archives.
After this operation, 13.2 MB of additional disk space will be used.
Get:1 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.33/deb cri-tools 1.33.0-1.1 [17.3 MB]
Get:2 https://ftp.gwdg.de/pub/openSUSE/repositories/isv%3A/cri-o%3A/stable%3A/v1.33/deb cri-o 1.33.0-2.1 [21.6 MB]
Get:3 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.33/deb kubectl 1.33.1-1.1 [11.7 MB]
Get:4 https://prod-cdn.packages.k8s.io/repositories/isv:/kubernetes:/core:/stable:/v1.33/deb kubelet 1.33.1-1.1 [15.9 MB]
Fetched 66.5 MB in 1s (76.9 MB/s)
...
```

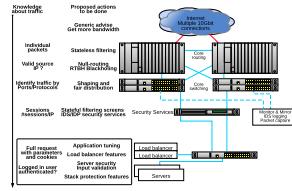
- I have had very little trouble with the CRI-O and K8s software installation and maintainance

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 CRI-O
- Smallest Kubernetes: Minikube - I run this on Debian
- Production deployment probably would use better tools like 🔐 Kubeadm
- Cilium for the networking part took some setup, quite a lot! Afterwards has been very easy
- Switching from NGINX Ingress to Gateway API with Cilium was well documented, even though a new feature

Conclusion Kubernetes on-premise



- It is hard getting started, took me a lot of YAML 😊
- Dont be discouraged – just learn some Linux, Debian, K8s, Cilium, BGP, NFS, YAML, NGINX, Git, Ruby, Jekyll and you are almost done
- 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-6a74857cc74c>

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

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



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/>