Certified Kubernetes Security Specialist (CKS) Handbook

Essential Guide to Exam Success

First Edition

Puru Tul⎈dhar

WHO IS THIS BOOK FOR?

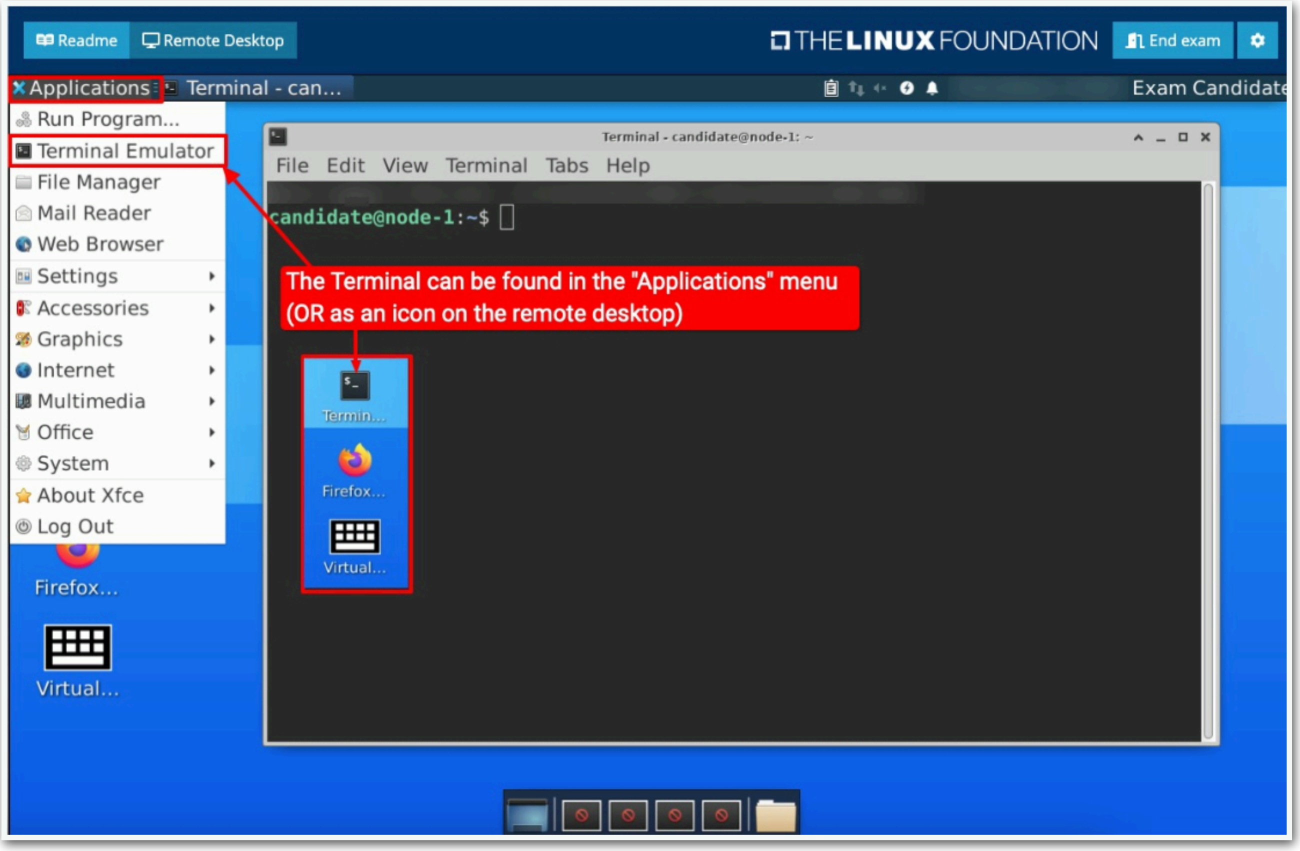
This book is for anyone committed to passing the CKS exam. Whether you're just beginning your preparation or need a final boost, this book is here to guide you every step of the way.

While there are plenty of resources out there, this book brings something special to the table. It's designed to give you that *extra edge* in passing the CKS exam. I understand the challenges of this certification, and the more you sharpen your skills, the greater your chances of success. That's why I wrote this book, to help you succeed.

Happy studying, and best of luck on your journey to success! I'm confident this focused handbook will be a valuable asset in your CKS preparation.

Puru Tuladhar
CKA & CKS Certified

TABLE OF CONTENTS



```
candidate@node-1:~$
```

The Terminal can be found in the "Applications" menu (OR as an icon on the remote desktop)

Fig: Terminal Emulator

> 💡 TIP: Use Linux keyboard shortcuts to Copy (Ctrl+Shift+C) & Paste (Ctrl+Shift+V) 📋 within the Linux Terminal.

1.3. 🦊 Firefox

Firefox is the default browser in the remote desktop, so make sure to use Ctrl+F (Find in Page) to quickly locate information in documentation. This will save you valuable time compared to scrolling endlessly.

1.4. 📋 Clipman

Clipman is a Linux clipboard manager available in the "Applications > Accessories" menu.

You can access it from the menu bar to browse your clipboard history while copying and pasting during the tasks. This will definitely save you time and boost your productivity during the exam.

> 🏷️ NOTE: Clipman is typically pre-activated. Look for the 📋 icon in the Remote Desktop menu bar.

1.5. 🐭 Mousepad

Mousepad is a text editor available in the "Applications > Accessories" menu.

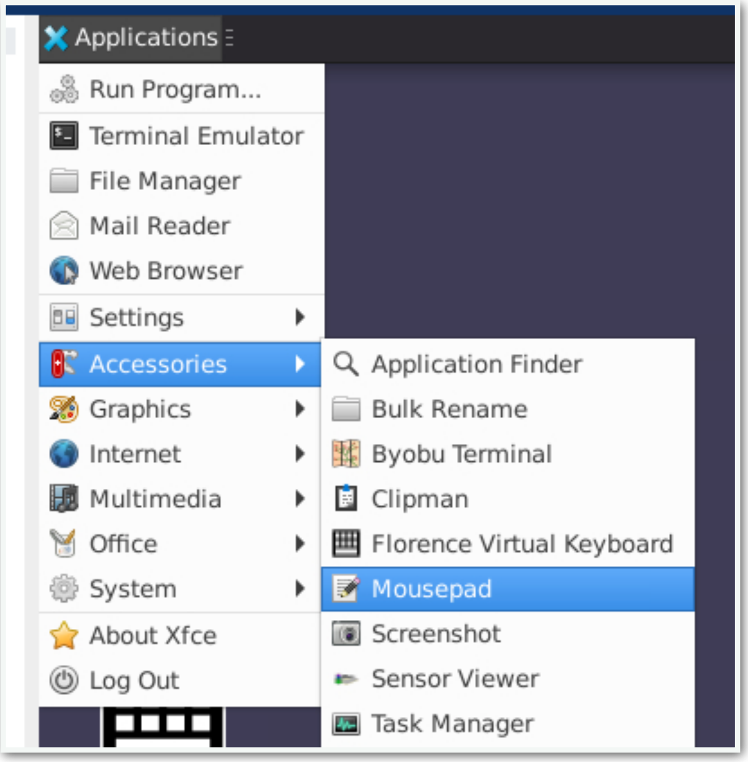


Fig: Mousepad & Clipman in Applications menu

Mousepad is super handy for jotting down notes or editing copied content during tasks. For example, you can prepare and save upgrade commands in Mousepad and easily copy and paste them when you need to execute on the nodes.

3. 🔎 Search Quickly

> ⚠️ IMPORTANT: Quickly searching documentation for relevant content is essential, allowing you to attempt all questions to maximize your score.

Searching quickly means efficiently finding the relevant content you need without wasting much time. With an average of 6 to 7.5 minutes per task and 16 to 20 long-form tasks to complete in 120 minutes, it's crucial to use the available documentation effectively.

> 🏷️ NOTE: If you get stuck on a task, don't panic; focus on completing as much of the task as possible. Time is tight, so the faster you locate the information, the better your chances of finishing all tasks within the allotted time.

📿 3-Steps Mantra for Quick Searches

1️⃣ Use short keywords (e.g: "kubectl quick")
2️⃣ Open the relevant page (e.g: "kubectl Quick Reference")
3️⃣ Use Find in Page in Firefox to get to the relevant section quickly.

Here are some common keywords I recommend to bring relevant pages to the top of search results. Ultimately, choose the ones that best help you find the information you need.

🏷️ Common Keywords

Short Keyword	Page
myuser.key	Certificates and Certificate Signing Requests
upgrade apply	Upgrading kubeadm clusters
supported field	Supported Fields for Conditions and Outputs
netpol	Network Policies
ingress tls	Ingress

4. 🐢 Shell Hacks

⚠️ IMPORTANT: Efficiently preparing and executing commands is often overlooked but is essential for smoothly completing tasks.

The exam is hands-on and involves extensive use of the Linux Terminal where you will be completing tasks. Quick shell navigation is essential for efficiently completing tasks. It's not only about knowing the commands but also about executing them swiftly and confidently.

💡 TIP: When tasked with creating and applying a manifest, a sample manifest file is usually provided at the end of the task. Be sure to use it.

🏷️ NOTE: During the exam, if you're unsure about any kubectl command, refer to the kubectl Quick Reference. It's a fast and reliable resource to guide you through syntax and options, helping you stay on track without wasting time.

1️⃣ Use kubectl alias and short names

```
k get ns     # namespace
k get ev     # events
k get cm     # configmaps
k get po     # pods
k get no     # nodes
k get svc    # services
k get deploy # deployment
k get ing    # ingress
k get netpol # networkpolicy
k get sa     # serviceaccount
```

⚠️ IMPORTANT: Make it a habit to use k instead of typing out kubectl. Trust me! small efficiencies like this add up and can save valuable seconds during the exam.

6. ⚙️ Cluster Setup – 10% 🏋️

6.1. 📝 Exam Curriculum

🔥 Network Policy	Use network security policies to restrict cluster level access.
🛡️ CIS Benchmark	Use CIS benchmark to review the security configuration of Kubernetes components (etcd, kubelet, kubedns, kubeapi)
🔀 Ingress with TLS	Properly set up Ingress objects with security controls.
🔒 Protect Metadata	Protect node metadata and endpoints.
🔎 Verify Platform Binaries	Verifying platform binaries before deploying.

6.2. 🔥 Network Policy

A network policy in Kubernetes acts as a firewall rule that defines what network traffic is allowed IN (ingress) or OUT (egress) for a specific set of Pods. Network policies are enforced and implemented by network plugin (CNI) such as Cilium.

By default, Kubernetes allows unrestricted pod-to-pod communication across the cluster, which can pose security risks if not properly managed.

> ⚠️ IMPORTANT: Network policy is namespace scoped.

> 🏷️ NOTE: During the exam, refer to the Network Policies documentation for ready-to-use policy templates and modify from there on.

1️⃣ In this policy, we block all incoming and outgoing traffic for Pods in the default namespace, with the exception of DNS traffic.

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: default-deny
  namespace: default
spec:
  podSelector: {}
  policyTypes:
  - Egress
  - Ingress
  egress:
  - to:
    ports:
    - port: 53
      protocol: TCP
    - port: 53
      protocol: UDP
```

6.5. 🔒 Protect Metadata

In cloud environments (such as AWS Instance Metadata or GCP VM Metadata), a metadata service is accessible from any VM and provides important information about the node, but it can also reveal sensitive data.

By default, all Pods running on a node (VM) in a Kubernetes cluster can access the metadata service. To enhance security and protect sensitive information, it's best practice to restrict Pod access to the metadata service using Network Policies.

1️⃣ In this policy, we block outgoing traffic from all Pods in the metadata namespace to the metadata service (169.254.169.254), while still permitting all other outgoing traffic.

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: metadata-deny
  namespace: metadata
spec:
 podSelector: {}
 policyTypes:
 - Egress
 egress:
 - to:
   - ipBlock:
       cidr: 0.0.0.0/0
       except: 169.254.169.254/32
```

9.4. 🚪 OPA Gatekeeper

OPA Gatekeeper is a CNCF graduated project that integrates OPA (a general-purpose policy engine) with Kubernetes with additional features, such as:

1. Kubernetes-native CRDs i.e, ConstraintTemplate, Constraint
2. Audit functionality.

For installing OPA Gatekeeper, follow the official installation guide.

🎟️ Admission Controllers

In Kubernetes, Admission Controllers enforce policies on objects during create, update, delete operations. Admission controller is fundamental to policy enforcement, and there are two types:

1. Validating Admission Controllers: Validate requests and either allow or reject them without modifying the request.
2. Mutating Admission Controllers: Modify requests before they are persisted in the database (i.e, etcd).

OPA Gatekeeper as an validating admission controller can for example:

- Require container images come from pre-approved image registry.
- Require specific labels on all resources.

OPA Gatekeeper as a mutating admission controller can:

- Inject sidecar containers into Pods.
- Set specific annotations on all resources.

10.5. 🖨️ Scan Images

Scanning container images is essential for detecting vulnerabilities before they reach the production environment. Integrating image scanning into the CI/CD pipeline further strengthens security by identifying these vulnerabilities early, a practice commonly known as "shift left" in the development lifecycle.

🛡️ Trivy

Trivy is the most popular open source security scanner. It can be used to scan container images for known CVEs and provides detailed reports that categorize vulnerabilities by their severity levels.

> 🏷️ NOTE: During the exam, the Trivy binary will be pre-installed on the cluster nodes.

1️⃣ Trivy Essential Commands

```
# Scan a container image
$ trivy image nginx:latest

# Filter for high or critical vulnerabilities
$ trivy image --severity HIGH,CRITICAL nginx:latest
```

> 💡 TIP: In the exam, you may be tasked with running a Trivy scan on the images of all pods in a specific namespace.

11.2. 📊 Behavioral Analytics

Behavioral analytics is the practice of collecting, analyzing, and interpreting data on the actions and behaviors of users or entities within a digital environment.

In Kubernetes, behavioral analytics can include observing container processes for system calls and file activities, to detect unusual or potentially malicious behavior

☎️ System calls

A system call is how a user-space process interacts with the Linux Kernel, for example requesting memory with mmap(). Allowing unrestricted system call access can be risky, as calls like execve are often linked to malicious actions.

System hardening tools like Seccomp and AppArmor help mitigate this risk by controlling and restricting the system calls a process can make, thus enhancing security.

1️⃣ Determine system calls made by an nginx container

```
# Retrieve the container ID
$ k get po nginx -o yaml | grep containerID

# Identify the node where the pod is running
$ k get pods nginx -o wide

# On the node, use crictl to get the container’s process ID
$ crictl inspect $CONTAINER_ID | grep pid

# On the node, use strace to trace system calls. Hit CTRL-C after a few
seconds to stop.
$ strace -f -cw —p $PID_ID
```

11.4. 🪵 Audit Logs

Audit logs in Kubernetes capture a detailed record of all HTTP requests made to the Kubernetes API server.

🪜 Request Stages

When Audit is enabled, each requests goes through the following stages and each stage generates an audit event and is recorded.

Stage	Description
RequestReceived	First audit request is received.
ResponseStarted	This stage is only generated for long-running requests (e.g. watch).
ResponseComplete	The response body has been completed and no more bytes will be sent.
Panic	Events generated when a panic occurred.

📄 Audit Policy

Audit policy defines rules about what events should be recorded and what data they should include. There are four levels to decide how much information should be recorded and stored:

Level	Action
None	Don't log events that match this rule.
Metadata	Log request metdata
Request	Log event metadata and request body but not response body
RequestResponse	Logs metadata, requests and response body.