Red Hat

Whitepaper

Certified Kubernetes Security Specialist study guide

Introduction

The Cloud Native Computing Foundation (CNCF) announced that the Certified Kubernetes Security Specialist (CKS) certification is now generally available. This guide provides a starting point to understanding the exam structure, topics, and exam-taking best practices. The guide will not provide direct questions found on the exam.

Let us start by reviewing the exam structure by referring to the Linux® Foundation documentation.

Exam structure and requirements

The CKS exam structure and requirements:

- 2 hours in duration.
- Requires a passing score of 67%.
- Contains 15-20 performance-based tasks.
- Uses Kubernetes version 1.19.
- Costs US$300.
- Offers a free exam retake.
- Includes a 2-year certification.
- Includes 12-month exam eligibility.
- Requires a valid Certified Kubernetes Administrator (CKA) certificate.

The format of the CKS exam is almost identical to the CKA exam, which helps if you are taking both tests in succession.

Exam format

The Linux Foundation has also outlined the exam format, and it is worth reading to help with exam expectations:

- Each task on this exam must be completed using a designated cluster/configuration context.
- 16 clusters comprise the exam environment, one for each task. Each cluster is made up of one supervisor node and one worker node.

- At the start of each task, an infobox provides you with the cluster name/context and the supervisor and worker node's hostname.
- You can switch the cluster/configuration context using a command such as: `kubectl config usecontext`.
- Nodes making up each cluster can be reached via ssh, using a command such as `ssh`.
- You don't have elevated privileges on any node by default, so there is no need to assume elevated privileges.
- You must return to the base node (hostname CLI) after completing each task.
- Nested-ssh is not supported.
- You can use `kubectl` and the appropriate context to work on any cluster from the base node. When connected to a cluster member via ssh, you will only work on that cluster via `kubectl`.
- Further instructions for connecting to cluster nodes will be provided in the appropriate tasks.
- The CKS environment is currently running Kubernetes v1.19. (Quarterly exam updates are planned to match future Kubernetes releases.)

Adding to the Linux Foundations format, there are a couple of extra notes to give more context:

- Vi/Vim is the default editor for the exam. If you require Gedit, Emacs, or Nano, it will have to be installed in each node you work on.
- Vi/Vim is preconfigured for the correct tabs/spaces required to format YAML files.

With that context, it is time to discuss how to prepare for the exam.

Preparing for the exam

To help prepare for the CKS exam, StackRox (now owned by Red Hat) has created a GitHub Repo that will create a Kubernetes cluster using version 1.19 and provide Kubernetes security tasks to evaluate your expertise. The repo also contains a full library of resources to help guide you through each section of the exam and outline the core Kubernetes security topics.

Getting started

- Register for the CKS at the Linux Foundation website.
- Follow the guidelines to book your exam, and set a date one to three months in advance.
- Read the complete Certified Kubernetes Security Specialist study guide.
- Follow the Kubernetes Security Specialist study guide GitHub repository and review the other resources outlined in the README file to better understand various concepts.

General tips for the exam

With lessons learned from the CKA exam, here are some general tips and resources for dealing with the performance- and task-based Kubernetes exams.

- Always review the Linux Foundation documentation for frequently asked questions and any updates to the exam structure.

- Read each question carefully and manage the time accordingly. If a question asks for a pod, make sure to create a pod and not a deployment. The exam will evaluate on the basis of the outputs to files and deployment/pod names. Errors in filenames or pods names might cause the response to be marked incorrect.
- Bookmark and use any resources from these domains and their respective subdomains:
  - Kubernetes documentation and resources
    - Kubernetes documentation
    - Kubernetes GitHub
    - Kubernetes blog
  - Tools
    - Trivy documentation
    - Sysdig documentation
    - Falco documentation
    - AppArmor documentation
- Recording task progression will help to prioritize where to spend the final moments during the exam. In previous exams, a notepad was provided during the exam. In the new format, there is a progress bar and the ability to flag a question. You can use the flag to signal that you are completely finished with a question or to indicate you are unsure of its answer so you can revisit it later.
- Because time is of the essence, take advantage of the kubectl cheat sheet, and use aliases to cut down on kubectl typos.
- Be proficient in vi/vim for file editing during the test. Luckily, the spacing is preformatted for YAML files.
- Pay attention to the question context. There will be a context change command at the beginning of every question. Since there are 16 unique clusters, make sure to address the correct cluster to use time efficiently (and produce fewer headaches).
- Never write a YAML file from scratch. Use the `-dry-run=client -o yaml > example.yaml` to output example formats without submitting the commands to the cluster. With the new format, the YAML files are given with their bare bones formatting. It will be up to you to understand how to populate and apply them. Use the files available as they will give you a better understanding of what is required of the question.
- Learn how to sort through JSON outputs. There might be a question where you are required to search through active pods/deployments for labels, memory limits, CPU limits, etc. You can save a significant amount of time on a low-value question by sifting through objects efficiently.
- The exam will use kubeadm for creating the Kubernetes cluster. Review the systemd basics, and review where the cluster configuration yaml files are located.
- When using ssh, ensure that you `exit` from the node to the `central node`.

The Linux Foundation is strict on the exam requirements and calls for an empty room, no visible writing materials, and a clean desk. Make sure to get a good night's sleep, drink some water beforehand, take 30 minutes before the test to get up, walk around, and take a few deep breaths.

During the exam

The exam lasts for two hours, although there will be about 15 minutes of setup with the proctor before the exam starts. Once the exam starts, take your time in the first couple of minutes to get settled with the test format. You are allowed two open windows, one for the exam and the other for using external pages. Aim to complete questions in 5 minutes on average because there are approximately 15 questions and 120 mins to complete the exam. This will give you some time for review at the end of the test.

The CKS, and Kubernetes exams in general, reward precision and time control. Rushing through the questions might give you more time for review; however, a misconfiguration or typo in the YAML files could lead to a loss of all points on a problem. It is better to be efficient the first time than to switch contexts multiple times. Lastly, if you have no idea what the question is asking, we recommend moving on to the next question. It is better to get a couple of questions answered correctly at the beginning than wasting time on unknowns.

Now that you understand the expectations, let us walk through the core exam concepts and topics.

Section 1: Cluster setup

The first section focuses on controlling access to the Kubernetes cluster. The Linux Foundation course outline highlights these six core concepts:

1. Use network policies to restrict cluster-level access.
2. Use the CIS Kubernetes Benchmark to review the security configuration of Kubernetes components (etcd, kubelet, kube-dns, kube-api).
3. Properly set up ingress objects with security control.
4. Protect node metadata and endpoints.
5. Minimize use of, and access to, GUI elements.
6. Verify platform binaries before deploying.

This section makes up 10% of the point total, and it is reasonable to assume there will be two or three questions about cluster setup.

Use network policies to restrict cluster-level access

Network access between Kubernetes pods is open internally by default. A significant security risk associated with this setup is a container being able to access and connect to other workloads within the cluster network. Network policies are the answer to this core vulnerability, which means network policies will, without a doubt, be on the exam.

Network policies got an update in version 1.19, so it is worth reviewing the functionality and YAML structure changes. Also, make sure to bookmark these resources as they will come in handy for reference during the exam. If you'd like to learn more about network policies, Calico has some great demos, and Viswajith Venugopal has provided an extensive writeup of ingress and egress network policies. Lastly, our GitHub repo has multiple questions focused on default-deny policies and setting up ingress objects.

Use the CIS Kubernetes Benchmark to review the security configuration of Kubernetes components (etcd, kubelet, kube-dns, kube-api)

This topic is challenging to narrow down into specific questions. It requires knowledge of the CIS benchmarks that cover version 1.16 to 1.18. An open source tool, kube-bench, runs validation of the Kubernetes components using the CIS benchmarks. It is worth knowing before the test; however, a question about kube-bench should not be on the test because the webpage is not a listed resource that can be used during the test. With the exam being created with Kubeadm, there might be questions focusing on fixing the core component config files such as `kube-apiserver`. Knowing how to configure and secure the Kubernetes components is vital to using functionality such as admission controllers, role-based access control (RBAC), and avoiding a setup where `--anonymous-auth` was set to `true`.

Properly set up ingress objects with security control

Kubernetes ingress is another concept that will be on the test. Questions could include adding TLS to a previous ingress object or setting up an IngressClass. IngressClass was introduced in version 1.18 and helped to specify how different controllers should implement ingress objects. There are 3 types of ingress controller setups to be aware of:

1. Cluster-wide ingress controller (default)
2. Single-namespace ingress controller
3. Ingress controller for specific ingress class

Be sure to understand how to implement TLS in Ingress objects and how to set up ingress objects and IngressClass objects.

Protect node metadata endpoints

The topic says it all here. Protecting node metadata and endpoints is always a top concern. In previous Kubernetes versions, the kubelet had a read-only port, port `10255`, that could be exploited to learn more about the pods running on the worker nodes. There could be a question where you have to reconfigure kubelet and disable the read-only port. However, the read-only capability has been deprecated, so it seems unlikely this would come up. It is worth knowing all of the endpoints that are required for the Kubernetes cluster to function.

Another use case might be a simple service check. There might be multiple services set up on the exam cluster, and then it is up to you to weed out unnecessary services and remove them. It will be worth brushing up on kubectl filtering capabilities and searching by spec.

Minimize use of, and access to, GUI elements

Similar to the previous concept, minimizing access to graphic user interface (GUI) elements is a significant security concern. This topic is fueled by previous security hacks that exposed the Kubernetes Dashboard to the public. To combat this issue, admins should set up internal-only facing dashboards with specific user access outlined in their configuration files.

There might be a question on the exam that requires changing NodePort services or proxying to a dashboard. Another option could be minimizing the permissions of a dashboard within the cluster. Regardless, making sure that GUI elements are secure should always be a top priority during cluster setup.

Verify platform binaries before deploying

Kubernetes binaries can be verified by referring to their specific checksum in GitHub. Because exam takers will have access to the Kubernetes GitHub repository, it is worth bookmarking the release section and understanding how to verify binary SHA256 hashes.

Section 2: Cluster hardening

The second section focuses on controlling access to the Kubernetes cluster environment. The Linux Foundation highlights these four core concepts in their course outline:

1. Restrict access to Kubernetes API.
2. Use RBAC to minimize exposure.
3. Exercise caution in using service accounts (e.g., disable defaults, minimize permissions on newly created ones).
4. Update Kubernetes frequently.

Knowing that this section makes up 15% of the point total, it is reasonable to assume there will be two or three questions about cluster hardening.

Restrict access to Kubernetes API

Restricting access to the API server is about 3 things:

1. Authentication.
2. Authorization.
3. Admission control.

The Kubernetes documentation outlines these topics well, and they are a recommended place to bookmark for the test. Restricting access to the Kubernetes API server is, and will remain, a prevalent topic that will re-emerge in various concepts throughout the test.

Starting with authentication, the CKS might contain a question on user and service account creation and might include creating user certifications or service accounts for deployments. The bootstrap tokens feature probably will not be used due to the limitations of the environment setup.

When it comes to authorization, the CKS will focus on RBAC configuration within the cluster as it is enabled by default today. However, there are other authorization modes to be aware of, including:

- Node authorization.
- Attribute-based access control (ABAC).
- Webhooks.

With the time limitation, the questions around authorization will most likely focus on implementing RBAC policies and using auth can-i to determine API access.

Lastly, admission control will continuously be in use throughout various CKS exam topics. An admission controller intercepts requests to the Kubernetes API after the request is authenticated and authorized but before the object is saved in the key-value store. Know the default admission controllers in the current version, and we recommend bookmarking and getting to know each controller intimately. A significant amount of this exam will be working with various admission controllers to secure the cluster. They will be highlighted as this study guide moves through the sections.

Use RBAC to minimize exposure

This section is somewhat of a repeat of the previous concept, except that it focuses exclusively on RBAC. The concepts include:

- Roles.
- ClusterRoles.
- RoleBindings.
- ClusterRoleBindings.

This concept will also highlight the binding of roles to "subjects" such as users, groups, and service accounts. Expect questions focused on binding service accounts and users to specific access within the cluster.

Exercise caution in using service accounts (e.g., disable defaults, minimize permissions on newly created ones)

This concept expands on the previous one and focuses on the proper implementation of subjects. This includes setting default service accounts with the lowest permissions and removing unnecessary service account permissions and using the `auth can-i` functionality to assess API access.

Update Kubernetes frequently

The last topic was added during the detailed CKS announcement and is ambiguous about how this will be tested. There might be an upgrade question as the documentation about upgrading with kubeadm has been significantly better in recent releases. For instance, you must upgrade from version 1.18 to 1.19 or possibly drain and update a single node on the cluster. This topic addition is most likely due to version 1.15 being the average cluster version in production today and, in parallel, the community's desire to get users to take advantage of the updated security features in the last few releases.

Section 3: System hardening

The third section of our study guide focuses on minimizing the attack surface in the cluster as well as kernel access. The Linux Foundation highlights these four core concepts in their course outline:

1. Minimize host OS footprint (reduce attack surface).
2. Minimize identity and access management (IAM) roles.
3. Minimize external access to the network.
4. Appropriately use kernel hardening tools such as AppArmor or seccomp.

This section makes up 15% of the point total, and it is reasonable to assume there will be three or four questions about system hardening.

Minimize host OS footprint (reduce attack surface)

Minimizing the surface area of attack on your workloads is always an important task. There are 3 main aspects to reducing the attack surface of your machines:

1. Removing unnecessary packages.
2. Identifying and addressing open ports.
3. Shutting down any unnecessary services.

When applying this to the CKS exam, it is improbable that you will have to navigate the Ubuntu OS and remove packages during the exam. Instead, the CKS might ask you to stop containers running with privileged permissions in the cluster. Also, CronJob can pose a serious threat around persistence and container exploitation that the CKS might try to highlight as well.

Network policies are the default network segmentation tool in Kubernetes. It is unlikely that you will have to use a tool like ufw to secure the host, although do not rule it out. Most likely, you will have to shut down exposed services and set up default deny rules inside Kubernetes namespaces to minimize network access.

The exam might also ask you to use admission controllers to limit what can and cannot be run in the cluster. Security contexts are used for multiple security aspects, such as setting process UIDs and not allowing write access to the container filesystem. Controllers such as `SecurityContextDeny` are useful tools for limiting the scope of pod processes as well.

Minimize IAM roles

Typically, identity and access management (IAM) roles are referenced when talking about cloud providers. Although it is not apparent how this concept will manifest during the exam, you can be sure that minimizing access through RBAC will be a consistent theme.

Minimize external access to the network

Minimizing external access is a slight repetition of the first concept. You might see the network policy implementation expand to include IP blocks and specific ingress and egress rules. Also, you might be required to investigate PodSecurityPolicies (PSP) that allow access to the host network or other privileges that might give a container elevated access. However, implementing PSPs during the exam is unlikely, considering that the feature will most likely be deprecated in version 1.21.

Appropriately use kernel hardening tools such as AppArmor and seccomp

The documentation for the CKS changes frequently, and as of this writing, it allows you to access AppArmor documentation during the exam. There is no link to seccomp documentation, but seccomp profiles have been a GA feature since 1.19. There is concern over how these concepts will be implemented during the exam. There is a lot of documentation to sift through, and a question on AppArmor implementation might be a time sink if you are not careful. Most likely, there will be a question where you will implement a pre-configured AppArmor profile on the host or using an annotation.

Section 4: Minimize microservice vulnerabilities

The fourth section of our study guide focuses on minimizing microservice vulnerabilities and securing pods at runtime. The Linux Foundation highlights these four core concepts in their course outline:

1. Set up appropriate OS-level security domains using options such as pod security policies (PSP), open policy agent (OPA), and security contexts.
2. Manage Kubernetes secrets.
3. Use container runtime sandboxes in multitenant environments (e.g., gvisor, kata containers).

4. Implement pod-to-pod encryption using mTLS.

This section makes up 20% of the point total, and it is reasonable to assume there will be three to five questions about minimizing microservice vulnerabilities.

Set up appropriate OS-level security domains using options such as pod security policies, open policy agent, and security contexts

Overall, this section is the most vague regarding its concepts and what will be asked during the exam. As mentioned previously, pod security policies (PSPs) will be deprecated in Kubernetes version 1.21, and open policy agent (OPA) documentation is not listed as a resource used during the exam. Security contexts are here to stay, and you should be well-versed in their implementation. While PSPs and OPA are useful to know about, it is not clear how they will be incorporated into the exam. There will most likely be a question about implementing a PSP through RBACs.

Pod security policies

Even though PSPs will be deprecated, their functionality within Kubernetes clusters should be well understood. PSPs are a cluster-level resource that control a wide variety of items, from Linux capabilities to UIDs. Unfortunately, one criticism is their confusing application through RBAC and subtle loopholes that can be exploited. PSPs are applied to the user and any pods they create, which means you need to be familiar with cluster-level roles to users, groups, and service accounts. You must also be aware of the various controls under the PSP and use the `auth can-i` functionality to help debug any authorization issues.

Open policy agent

OPA has been integrated into the Kubernetes admission controller framework since version 3.0. The documentation is scant, so assume that if there is a question about it, it will be pulled directly from the blogs on kubernetes.io.

Security context

Security context refers to the `PodSpec` and the permissions associated with each pod. Contexts can be set at the pod and container level, and a question focused on debugging and identifying escalating privileges is a simple example that might be used. Overall, any questions around security contexts are going to tie into topics such as seccomp and AppArmor from the previous section and other privilege and access control settings.

Manage Kubernetes secrets

If you are taking the CKS, it means you have passed the CKA. You have already had to deal with secrets management from the first exam, so expect expanding that core concept here. Examples that focus on service account tokens or bootstrap token secrets seem like the next steps. As mentioned in this section, you can also guarantee that TLS secrets overlap with other core security concepts. Overall, focus on knowing how to implement secrets securely and ensure that other containers cannot access the secrets.

Use container runtime sandboxes in multitenant environments (e.g., gvisor, kata containers)

It is essential to understand container runtime sandboxes and their use cases. There might be a question on the exam asking you to implement a sandbox using the `runtimeClassName:` spec. The exam would have to provide you with gvisor or kata containers enabled in the cluster. The main issue is the lack of documentation from the list of sources you can reference during the exam.

Implement pod-to-pod encryption using mTLS

mTLS is a core concept to securing pod-to-pod communications. Although there might be an example that combines secrets, ingress, and mTLS into a single larger question. It is unlikely that the exam will ask you to create the certificates. However, it is worth bookmarking certificate signing requests and understanding how to implement kubeconfig access and mTLS authentication credentials.

Section 5: Supply chain security

The fifth section of our study guide focuses on supply chain security. The Linux Foundation highlights these four core concepts in its course outline:

1. Minimize base image footprint.
2. Secure your supply chain: put registries on the allowlist, sign, and validate images.
3. Use static analysis of user workloads (e.g., Kubernetes resources, Dockerfiles).
4. Scan images for known vulnerabilities.

This section makes up 20% of the point total, and it is reasonable to assume there will be three to five questions about supply chain security.

Minimize base image footprint

Regardless of how this is implemented on the test, minimizing your base images is always a good idea to decrease the attack surface of your containers. Always make sure only to include the packages that are necessary for each containerized application. When choosing a base image, note how well maintained the image is and its default installed software. On the exam, expect that you will have the option of selecting from a range of base images and choosing their defaults. There might be a question requiring Trivy to view CVEs related to a base image and then prioritize image selection accordingly. As a core concept, image scanning and minimizing your images is a handy way to lower the attack surface of your clusters.

Secure your supply chain: allow permitted registries, sign and validate images

Securing the images that are allowed to run in your cluster is essential. Also, you will need to verify that the pulled image is from the correct source. The `ImagePolicyWebhook` admission controller will allow you to set up rules around what images should be allowed within the cluster. An example rule the admission controller could monitor is not allowing any image with the tag `latest`. You will most likely have to connect the `ImagePolicyWebhook` with a previously set up webhook server during the exam.

Use static analysis of user workloads (e.g., Kubernetes resources, Dockerfiles)

Static analysis might be the most straightforward concept outlined in the Linux Foundation course outline. You will need to vet the configuration of Kubernetes YAML files and Dockerfiles and fix any security issues. This includes setting secure base images, removing unnecessary packages, stopping containers from using elevated privileges, and removing the ability to ssh into a container. When hardening Kubernetes resources, look for elevated privileges, security contexts that allow for a UID of 0, and host volumes that should not be mounted.

Scan images for known vulnerabilities

As mentioned in the previous section, there seems to be some crossover between these two topics. Out of the open source tools that are allowed, Trivy is the only one focused on container scanning. You are also allowed to use the GitHub documentation during the exam, so it's worth bookmarking the quick start documentation.

Section 6: Monitoring, logging, and runtime security

Our study guide's sixth and final section focuses on monitoring, logging, and runtime security within the cluster. The Linux Foundation highlights these six core concepts in its course outline:

1. Perform behavioral analytics on syscall process and file activities at the host and container level to detect malicious activities.
2. Detect threats within a physical infrastructure, apps, networks, data, users, and workloads.
3. Detect all phases of attack regardless of where it occurs and how it spreads.
4. Perform deep analytical investigation and identification of bad actors within the environment.
5. Ensure immutability of containers at runtime.
6. Use audit logging to monitor access.

This section makes up 20% of the point total, and it is reasonable to assume there will be three to five questions about monitoring, logging, and runtime security.

Perform behavioral analytics on syscall process and file activities at the host and container level to detect malicious activities

To perform behavioral analysis of syscall and file activities, you will need to implement a tool to detect threats. Falco is a CNCF incubating project listed in the course documentation as a resource available during the exam. Assume that you will have to use Falco to detect some malicious activity and output it to a file, similar to questions from the CKA.

Detect threats within a physical infrastructure, apps, networks, data, users, and workloads

This concept generalizes a lot of previous topics covered in past blogs. Assuming that each of the questions takes an average of five to six minutes, it is unlikely the exam will have complicated problems that cannot be solved relatively quickly. One possibility might require you to fully assess a deployment in the cluster and write any vulnerabilities that are found in a file. The exam seeks to determine your knowledge of Kubernetes security threats and how to mitigate them. This concept seems overly broad to narrow it down to any specific topic.

Detect all phases of attack regardless of where it occurs and how it spreads

This is another somewhat broad concept; however, it does highlight various methods of exploitation. Mounted volumes, downloading packages, or using malicious running containers on the host expose a significant attack surface that you need to be aware of. There is an excellent blog series focused on the MITRE ATT&CK Framework, which is a great resource for reviewing relevant threats for various attack phases.

Perform deep analytical investigation and identification of bad actors within the environment

This is another vague concept that encompasses a variety of techniques and topics that have been covered in previous sections. Anticipate questions that tell you to take action on something that you will have to correct. These questions might call for you to change config files, remove any misconfigurations, or protect secrets. Audit logging will help to identify bad actors and changes in Kubernetes environments as well.

Ensure immutability of containers at runtime

The principle of container immutability means that the containers you have deployed are never changed once they are running—only their images are updated. You want to ensure that the cluster's containers do not execute malicious code added through a download or mounted volume. You should also be aware of the sidecar pattern and how a volume can be mounted to both containers simultaneously.

Use audit logging to monitor access

Audit logging will make up a significant amount of this section's points. The audit policy feature in Kubernetes is another admission controller that allows users to specify what events should be recorded and what data is included. You will most likely need to set up an audit policy during the exam. This question might also tie into identifying bad actors as an audit policy is a helpful way to discover users who are making malicious or unqualified requests in the cluster. Lastly, the log backend might be used to output the audit logs to a specific file location.

Further reading: Implementing Kubernetes-native security with Red Hat

Security platforms built specifically to protect Kubernetes offer powerful security and operational advantages. Kubernetes-native security applies controls at the Kubernetes layer, ensuring consistency, automation, and scale. Organizations successfully deploy security as code, enabling security that's built in, not bolted on.

Read this *Kubernetes-native security: What it is and why it matters* whitepaper to learn about the key features and benefits of Kubernetes-native security and how it's different from existing container security approaches to deliver protections that are built specifically for Kubernetes environments.

North America
1 888 REDHAT1
www.redhat.com

Europe, Middle East, and Africa
00800 7334 2835
europe@redhat.com

Asia Pacific
+65 6490 4200
apac@redhat.com

Latin America
+54 11 4329 7300
info-latam@redhat.com

facebook.com/redhatinc
@RedHat
linkedin.com/company/red-hat