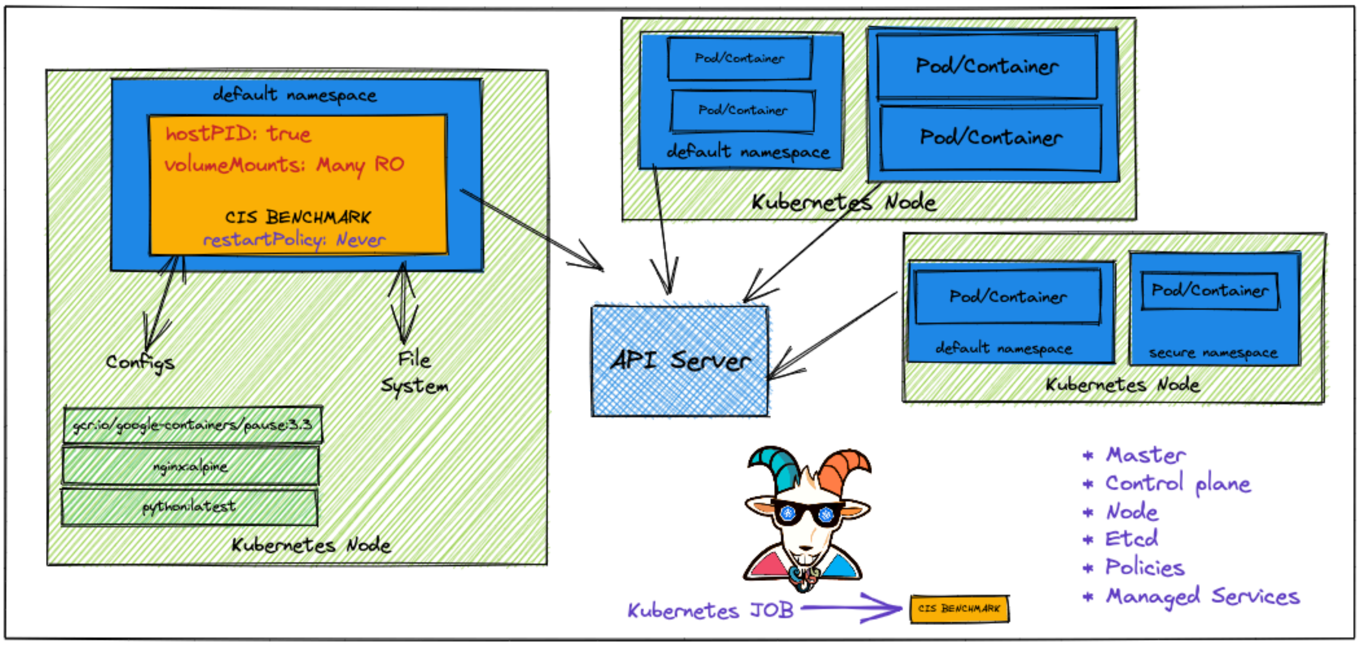
⎈ Kubernetes CIS benchmarks analysis

🙌 Overview

This scenario is very useful in performing Kubernetes security audits and assessments. Here we will learn to run the popular CIS benchmark audit for the Kubernetes cluster and use the results for the further exploitation or fixing of the misconfigurations and vulnerabilities. This is very important and mandates if you are coming from an audit and compliance background in the modern world of containers, Kubernetes, and cloud native ecosystems.



By the end of the scenario, we will understand and learn the following

1. You will learn to perform CIS benchmark audit for Kubernetes clusters
2. Working with jobs, Pods in Kubernetes, and other resources in the cluster
3. Gain visibility of the entire Kubernetes cluster security posture and understand the risks

⚡ The story

This scenario is mainly to perform the Kubernetes CIS benchmarks analysis on top of Kubernetes nodes and cluster resources to identify the possible security vulnerabilities.

- To get started with this scenario you can either access the node and perform by following kube-bench security or run the following command to deploy kube-bench as **Kubernetes job**

> INFO
>
> - To get started with the scenario, you can deploy the Kubernetes CIS benchmarks job using the following commands
>
> ```
> kubectl apply -f scenarios/kube-bench-security/node-job.yaml
> ```
>
> ```
> kubectl apply -f scenarios/kube-bench-security/master-job.yaml
> ```

🎯 Goal

> TIP
>
> The goal of this scenario is to perform the Kubernetes CIS benchmark audit and obtain the results from the audit.

Hints & Spoilers

▸ ✨ Not sure how to run the audit?

🎉 Solution & Walkthrough

🎲 Method 1

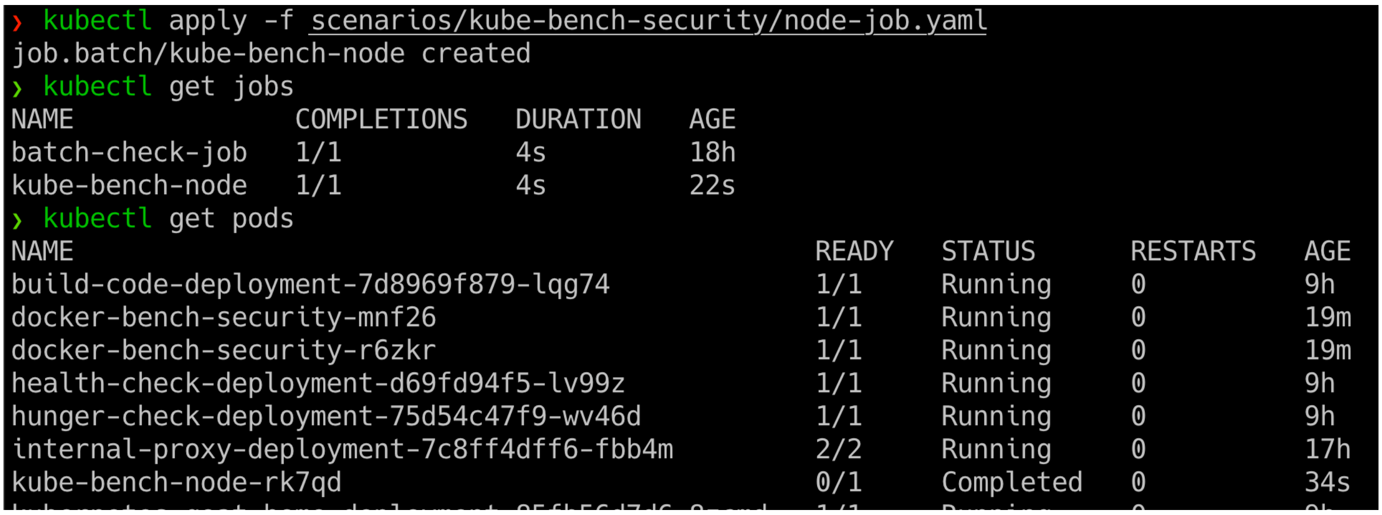
- We can deploy the Kubernetes CIS benchmarks by running the following command

```
kubectl apply -f scenarios/kube-bench-security/node-job.yaml
```

- Now we can obtain the list of jobs and associated pods information by running the following command

```
kubectl get jobs
```

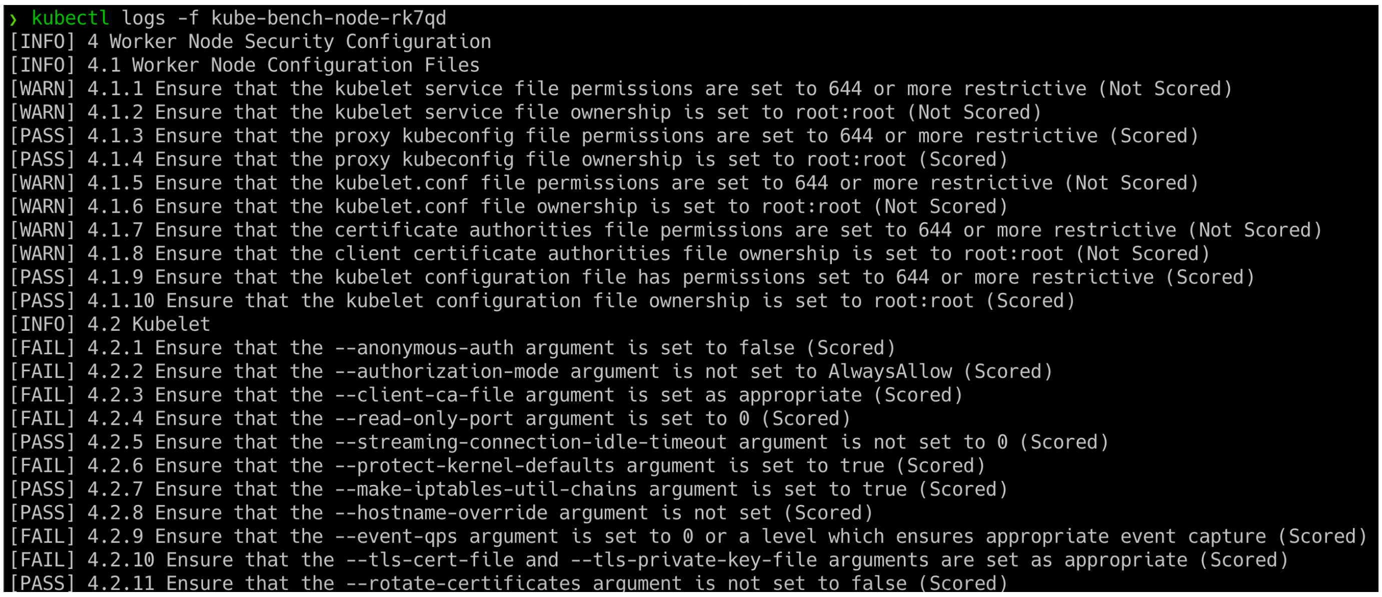
```
kubectl get pods
```



```
› kubectl apply -f scenarios/kube-bench-security/node-job.yaml
job.batch/kube-bench-node created
› kubectl get jobs
NAME              COMPLETIONS   DURATION   AGE
batch-check-job   1/1           4s         18h
kube-bench-node   1/1           4s         22s
› kubectl get pods
NAME                                          READY   STATUS      RESTARTS   AGE
build-code-deployment-7d8969f879-lqg74        1/1     Running     0          9h
docker-bench-security-mnf26                   1/1     Running     0          19m
docker-bench-security-r6zkr                   1/1     Running     0          19m
health-check-deployment-d69fd94f5-lv99z       1/1     Running     0          9h
hunger-check-deployment-75d54c47f9-wv46d      1/1     Running     0          9h
internal-proxy-deployment-7c8ff4dff6-fbb4m    2/2     Running     0          17h
kube-bench-node-rk7qd                         0/1     Completed   0          34s
```

- Once we have identified the pod, then we can get the audit results by running the following command. Make sure to replace the pod name in the following command

```
kubectl logs -f kube-bench-node-xxxxx
```



```
› kubectl logs -f kube-bench-node-rk7qd
[INFO] 4 Worker Node Security Configuration
[INFO] 4.1 Worker Node Configuration Files
[WARN] 4.1.1 Ensure that the kubelet service file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.2 Ensure that the kubelet service file ownership is set to root:root (Not Scored)
[PASS] 4.1.3 Ensure that the proxy kubeconfig file permissions are set to 644 or more restrictive (Scored)
[PASS] 4.1.4 Ensure that the proxy kubeconfig file ownership is set to root:root (Scored)
[WARN] 4.1.5 Ensure that the kubelet.conf file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.6 Ensure that the kubelet.conf file ownership is set to root:root (Not Scored)
[WARN] 4.1.7 Ensure that the certificate authorities file permissions are set to 644 or more restrictive (Not Scored)
[WARN] 4.1.8 Ensure that the client certificate authorities file ownership is set to root:root (Not Scored)
[PASS] 4.1.9 Ensure that the kubelet configuration file has permissions set to 644 or more restrictive (Scored)
[PASS] 4.1.10 Ensure that the kubelet configuration file ownership is set to root:root (Scored)
[INFO] 4.2 Kubelet
[FAIL] 4.2.1 Ensure that the --anonymous-auth argument is set to false (Scored)
[FAIL] 4.2.2 Ensure that the --authorization-mode argument is not set to AlwaysAllow (Scored)
[FAIL] 4.2.3 Ensure that the --client-ca-file argument is set as appropriate (Scored)
[FAIL] 4.2.4 Ensure that the --read-only-port argument is set to 0 (Scored)
[PASS] 4.2.5 Ensure that the --streaming-connection-idle-timeout argument is not set to 0 (Scored)
[FAIL] 4.2.6 Ensure that the --protect-kernel-defaults argument is set to true (Scored)
[PASS] 4.2.7 Ensure that the --make-iptables-util-chains argument is set to true (Scored)
[PASS] 4.2.8 Ensure that the --hostname-override argument is not set (Scored)
[FAIL] 4.2.9 Ensure that the --event-qps argument is set to 0 or a level which ensures appropriate event capture (Scored)
[FAIL] 4.2.10 Ensure that the --tls-cert-file and --tls-private-key-file arguments are set as appropriate (Scored)
[PASS] 4.2.11 Ensure that the --rotate-certificates argument is not set to false (Scored)
```

- Now based on the vulnerabilities you see from the Kubernetes CIS benchmarks, you can proceed with further exploitation

- Hooray 🥳 , now we can see that it returns the all security issues/misconfigurations from the cluster

References

- kube-bench
- CIS Benchmarks for Kubernetes

Edit this page