CIS benchmark with kube-bench

Kube-bench is an open-source tool that helps you assess the security of your Kubernetes cluster by running checks against the Center for Internet Security (CIS) Kubernetes benchmark. The CIS Kubernetes benchmark is a set of security best practices that can help you harden your Kubernetes cluster.

Overview kube-bench:
Kube-Bench: Kubernetes CIS Benchmarking Tool [Guide] (devopscube.com)
Overview cis benchmark:
https://www.cisecurity.org/benchmark/kubernetes

Specifying the benchmark or Kubernetes version

kube-bench uses the Kubernetes API, or access to the kubectl or kubelet executables to try to determine the Kubernetes version, and hence which benchmark to run. If you wish to override this, or if none of these methods are available, you can specify either the Kubernetes version or CIS Benchmark as a command line parameter.

You can specify a particular version of Kubernetes by setting the --version flag or with the KUBE_BENCH_VERSION environment variable. The value of --version takes precedence over the value of KUBE_BENCH_VERSION.

For example, run kube-bench using the tests for Kubernetes version 1.13:

```
kube-bench --version 1.13
```

You can specify --benchmark to run a specific CIS Benchmark version:

```
kube-bench --benchmark cis-1.5
```

kube-bench uses the Kubernetes API, or access to the kubectl or kubelet executables to try to determine the Kubernetes version, and hence which benchmark to run. If you wish to override this, or if none of these methods are available, you can specify either the Kubernetes version or CIS Benchmark as a command line parameter.

You can specify a particular version of Kubernetes by setting the --version flag or with the KUBE_BENCH_VERSION environment variable. The value of --version takes precedence over the value of KUBE_BENCH_VERSION.

For example, run kube-bench using the tests for Kubernetes version 1.17:

```
kube-bench --version=1.17.0
```

You can get current kubernetes version with command: kubectl version
And check Git Version value.
Note: It is an error to specify both --version and --benchmark flags together

Specifying Benchmark sections

If you want to run specific CIS Benchmark sections (i.e master, node, etcd, etc...) you can use the run --targets subcommand.

```
kube-bench run --targets master,node
```

Or

```
kube-bench run --targets master,node,etcd,policies
```

If no targets are specified, kube-bench will determine the appropriate targets based on the CIS Benchmark version and the components detected on the node. The detection is done by verifying which components are running, as defined in the config files (see Configuration.

Reference: https://github.com/aquasecurity/kube-bench/blob/main/docs/flags-and-commands.md

Kube-bench will run a series of checks against your Kubernetes cluster and report any security issues that it finds. You can use this information to harden your Kubernetes cluster and improve its security posture.

The mapping between kube-bench and CIS Benchmark versions is as follows:

https://github.com/aquasecurity/kube-bench/blob/main/docs/platforms.md#cis-kubernetes-benchmark-support

If you are running a newer version of Kubernetes, you can use the latest version of kube-bench. If you are running an older version of Kubernetes, you can use the corresponding version of kube-bench.

what about --check flag, difference between --benchmark flag and --check flag ?

The --benchmark flag specifies the CIS Kubernetes benchmark version that kube-bench should use. The --check flag specifies the individual checks that kube-bench should run.

For example, to run kube-bench against the CIS Kubernetes benchmark version 1.17.0 and only run the checks for authentication and authorization, you would use the following command:

```
kube-bench --benchmark=1.17.0 --check="1.1,1.2"
```

The --benchmark flag is required, but the --check flag is optional. If you do not specify the --check flag, kube-bench will run all of the checks in the CIS Kubernetes benchmark.
Here is a table that summarizes the difference between the --benchmark flag and the --check flag:
| Flag | Description |
--benchmark | Specifies the CIS Kubernetes benchmark version that kube-bench should use. |
--check | Specifies the individual checks that kube-bench should run. |

Now, I will try run kube-bench for our project with docker.

```
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v ~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run
```

This script runs kube-bench in Docker and checks whether Kubernetes is deployed securely by

running the checks documented in the CIS Kubernetes Benchmark[1]. The script runs kube-bench with the following parameters:
- `--pid=host`: This flag allows kube-bench to access the host's PID namespace.
- `-v /etc:/etc:ro -v /var:/var:ro`: These flags mount the host's `/etc` and `/var` directories as read-only volumes inside the container.
- `-v $(which kubectl):/usr/local/mount-from-host/bin/kubectl`: This flag mounts the `kubectl` binary from the host into the container.
- `-v ~/.kube:/.kube -e KUBECONFIG=/.kube/config`: These flags mount the `~/.kube` directory from the host into the container and set the `KUBECONFIG` environment variable to `/root/.kube/config`.
- `-t docker.io/aquasec/kube-bench:latest`: This specifies that kube-bench should be run using the latest version of the Docker image provided by Aqua Security.

And we have test results with each check list and Remediations

```
[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.1 Ensure that the API server pod specification file permissions are set to 644 or more restrictive (Automated)
[PASS] 1.1.2 Ensure that the API server pod specification file ownership is set to root:root (Automated)
[FAIL] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)

== Remediations master ==
1.1.12 On the etcd server node, get the etcd data directory, passed as an argument --data-dir,
from the below command:
ps -ef | grep etcd
Run the below command (based on the etcd data directory found above).
For example, chown etcd:etcd /var/lib/etcd
```

Now we will follow Remediations and run again to see the test results.

```
Useradd etcd
chown etcd:etcd /var/lib/etcd
docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v
~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run etcd --check 1.1.12

[INFO] 1 Master Node Security Configuration
[INFO] 1.1 Master Node Configuration Files
[PASS] 1.1.12 Ensure that the etcd data directory ownership is set to etcd:etcd (Automated)
```

and this time we passed. Now I will create custom script to test certain items for etcd, kubelet (node) and master. you can customize the checklists you want to test or simply use kube-bench run to run all the tests.

Create file cis-etcd.sh

```
#!/bin/bash
#cis-etcd.sh
#total_fail=$(kube-bench run --targets etcd  --version 1.15 --check 2.2 --json | jq .[].total_fail)
total_fail=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v
~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run ectd --version 1.20 --check 2.2 --json
| jq '.Totals.total_fail')
fail_result=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v
~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run ectd --version 1.20 --check 2.2)
if [[ "$total_fail" -ne 0 ]];
    then
        echo "CIS Benchmark Failed ETCD while testing for 2.2"
```

```
        exit 1;
    else
        echo "CIS Benchmark Passed for ETCD - 2.2"
        echo $fail_result
fi;
```

In this script the `--check` flag specifies which checks should be run. In this case, `--check 2.2` is used to check for compliance with CIS Benchmark 2.2 The latest version of kube-bench is v0.6.6.

The specified --targets "ectd" are not configured for the CIS Benchmark cis-1.20, so you just need use *kube-bench run ectd* without tag --target.

Similarly, we continue to create file cis-kubelet.sh

```
#!/bin/bash
#cis-kubelet.sh
#total_fail=$(kube-bench run --targets node  --version 1.15 --check 4.2.1,4.2.2 --json | jq .[].total_fail)
total_fail=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v ~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run --targets node --version 1.20 --check 4.2.1,4.2.2 --json | jq '.Totals.total_fail')
fail_result=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v ~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run --targets node --version 1.20 --check 4.2.1,4.2.2 )
if [[ "$total_fail" -ne 0 ]];
    then
        echo "CIS Benchmark Failed Kubelet while testing for 4.2.1, 4.2.2"
        exit 1;
    else
        echo "CIS Benchmark Passed Kubelet for 4.2.1, 4.2.2"
        echo $fail_result
fi;
```

Create file cis-master.sh

```
#!/bin/bash
#cis-master.sh
#total_fail=$(kube-bench master  --version 1.15 --check 1.2.7,1.2.8,1.2.9 --json | jq .[].total_fail)
total_fail=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v ~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run --targets master --version 1.20 --check 1.2.7,1.2.8,1.2.9 --json | jq '.Totals.total_fail')
fail_result=$(docker run --pid=host -v /etc:/etc:ro -v /var:/var:ro -v $(which kubectl):/usr/local/mount-from-host/bin/kubectl -v ~/.kube:/.kube -e KUBECONFIG=/.kube/config -t docker.io/aquasec/kube-bench:latest run --targets master --version 1.20 --check 1.2.7,1.2.8,1.2.9 )
if [[ "$total_fail" -ne 0 ]];
    then
        echo "CIS Benchmark Failed MASTER while testing for 1.2.7, 1.2.8, 1.2.9"
        exit 1;
    else
        echo "CIS Benchmark Passed for MASTER - 1.2.7, 1.2.8, 1.2.9"
        echo $fail_result
fi;
```

Add stage below into jenkins file

```
stage('K8S CIS Benchmark') {
    steps {
     script {
       parallel(
         "Master": {
           sh "bash cis-master.sh"
         },
         "Etcd": {
           sh "bash cis-etcd.sh"
         },
         "Kubelet": {
           sh "bash cis-kubelet.sh"
         }
       )
     }
    }
  }
```

Run build pipeline.