Amazon EKS as a fully capable platform for the CNTi Test Suite

RUNNING CNTI TEST SUITE V2.0.0 AGAINST THE COREDNS EXAMPLE CNF ON AMAZON EKS 1.36

Communications service providers increasingly ask their suppliers for CNTi Test Suite results. The suite runs against a deployed cloud native network function (CNF) and needs a multi-node Kubernetes cluster with the containerd runtime, admin access, a default StorageClass, a metrics pipeline, and outbound access for its tooling. Amazon Web Services, in collaboration with the Cloud Native Telecom Initiative (CNTi), set up an Amazon Elastic Kubernetes Service (Amazon EKS) 1.36 environment that provides all of these. On it, the team installed the suite's CoreDNS example CNF and ran the Observability & Diagnostics and Security categories of CNTi Test Suite v2.0.0, with every tool version pinned and the results packaged for review.

PARTNERS Amazon Web Services · Cloud Native Telecom Initiative (LF Networking)

TEST SUITE CNTi Test Suite v2.0.0 (commit 8f42cb8)

PLATFORM Amazon EKS 1.36 · Amazon Linux 2023 · containerd 2.2.7

CNF UNDER TEST
CoreDNS, the suite's example CNF

CNTi CATEGORIES RUN Observability & Diagnostics · Security (2 of 7)

CNF SECURITY 13 of 19 tests passed · 6 findings with remediation

KUBERNETES CONFORMANCE
451 passed · 0 failed (Sonobuoy)

WORKER RECOVERY
4 of 4 nodes returned Ready

CHALLENGE

CNTi Test Suite results are becoming a routine request from CSPs to their CNF suppliers. The suite's documentation uses kind for a first run, which is quick to set up but runs on a single host. That leaves node-level behaviour untested and says little about the infrastructure the CNF will run on in production.

Two further issues make results hard to use. A run whose tool versions and cluster configuration were not recorded cannot be reproduced or compared with a later one. And a single total score does not show which tests ran, which were skipped, or whether a finding concerns the CNF or the cluster underneath it.

The question for this work was whether a managed Kubernetes service can provide everything the suite needs, and whether the output can be recorded clearly enough for a CSP to review.

SOLUTION

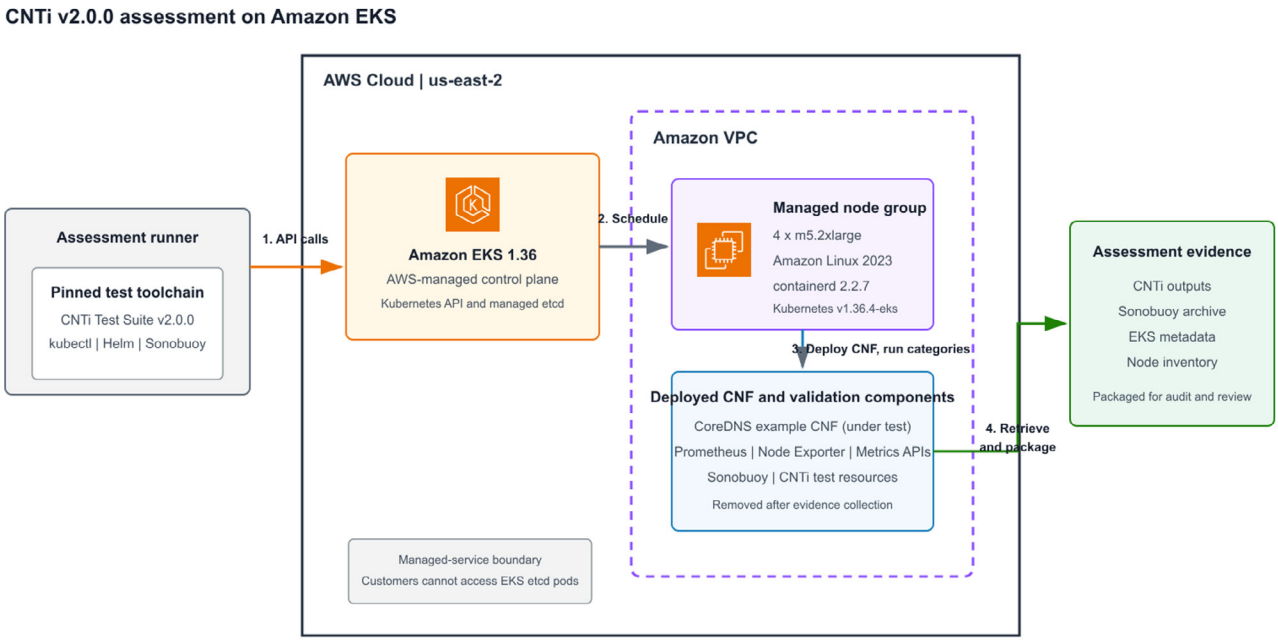
The run follows the suite's standard sequence on Amazon EKS: cnti-testsuite setup to prepare the suite, cnti-testsuite cnf_install with the CoreDNS example configuration, and then the selected test categories. A wrapper script drives each step, adds the two independent checks, and confirms before each step that the earlier ones completed against the same cluster. In the order they ran:

STEP	TOOL	WHAT IT TESTS
Create the cluster, build the runner, run setup	eksctl, pinned runner image, CNTi	Prepares the environment
Install the CNF, run Observability & Diagnostics	CNTi `cnf_install, observability --poc`	The CNF
Kubernetes conformance baseline	Sonobuoy, independent of the suite	The cluster
Remove Sonobuoy, run Security	CNTi `security`	The CNF
Reboot a worker and verify recovery	Amazon EC2 reboot API, the team's own check	The nodes
Package the evidence	Wrapper script, SHA-256 checksum	Records the run

SOLUTION *(cont.)*

Sonobuoy creates temporary ClusterRoles and ClusterRoleBindings. The script deletes Sonobuoy before the Security category runs and stops if any of those objects remain, so the security scan sees only the CNF and the cluster's own configuration.

The runner is a container image with CNTi pinned to commit 8f42cb8 and the AWS CLI, kubectl, Helm and Sonobuoy each pinned to a version and checked against a recorded SHA-256 value. Before any test runs, the script compares the versions inside the built image with the requested ones.



CNTi Test Suite v2.0.0 on Amazon EKS

The runner calls the AWS and EKS APIs; the EKS control plane schedules the CoreDNS CNF and the temporary test components onto a four-node managed node group; results are collected and packaged.

HOW AMAZON EKS MEETS THE SUITE'S REQUIREMENTS

REQUIREMENT	HOW IT WAS MET ON AMAZON EKS
Conformant Kubernetes API on containerd	Amazon EKS is a CNCF Certified Kubernetes product; Amazon Linux 2023 managed nodes run containerd
Several schedulable nodes	A fixed four-node managed node group (`m5.2xlarge`), autoscaling off so the topology stays known
Admin access	EKS access entries; for this short-lived cluster, the cluster creator identity
Default StorageClass	The `aws-ebs-csi-driver` add-on with `defaultStorageClass` enabled, declared in the cluster manifest
Resource and custom metrics APIs	The `metrics-server` add-on, plus Prometheus and Prometheus Adapter installed for the run and checked Available before testing
Outbound access for charts and images	Default managed node group networking
A way to restart a node	The Amazon EC2 reboot API, using the instance ID from the Kubernetes node's provider ID

THE ROLE OF OPEN SOURCE AND CNTi

Apart from the managed service itself, every component in the run is open source: the CNTi Test Suite, Sonobuoy, CoreDNS, Prometheus, and the Helm charts used to install them. Because Amazon EKS is CNCF Certified Kubernetes, the suite works against the standard Kubernetes API with no EKS-specific changes.

Three aspects of CNTi Test Suite v2.0.0 shaped how the results are reported. The suite evaluates a deployed CNF; v2.0.0 retired the separate platform test suite, so questions about the cluster are answered by other means, here Sonobuoy and the EKS certification, and reported separately. Each test ends with an explicit status: passed, failed, skipped, na or error. And the exit code distinguishes a run in which a test failed (1) from one in which the suite itself errored (2) or the command line was invalid (64), so the script treats a failed test as a valid result rather than a broken run.

The CoreDNS example ships with the suite and installs in a few minutes, which makes it a practical first workload for checking a new environment before moving to a full network function.

RESULTS

CHECK	RESULT	APPLIES TO
Security category	13 of 19 tests passed, 6 failed (635 of 745 points)	The CoreDNS CNF
Observability & Diagnostics category	`log_output` (essential) passed; `prometheus_traffic` failed; `open_metrics, routed_logs, tracing` skipped	The CoreDNS CNF
Kubernetes conformance	451 tests passed, 0 failed; node log collection passed on all 4 nodes	The cluster
Worker recovery	Boot ID changed after reboot; the test workload and 4 of 4 nodes returned Ready	The nodes
Reboot a worker and verify recovery	Amazon EC2 reboot API, the team's own check	The nodes
Package the evidence	Wrapper script, SHA-256 checksum	Records the run

The six security failures are `privilege_escalation, non_root_containers, linux_hardening, ingress_egress_blocked, immutable_file_systems` and `service_account_mapping`. Each names the affected resource and a remediation. They come from the example chart's manifests, which do not set `runAsNonRoot, allowPrivilegeEscalation: false` or a `seccompProfile`, and do not include a NetworkPolicy. All six can be fixed in the chart's Helm values, and none concerns Amazon EKS.

In Observability & Diagnostics, CoreDNS passed the essential logging test. The remaining four are bonus tests that look for optional integrations. The example deployment includes no Prometheus scrape configuration, log router or tracing backend, so `prometheus_traffic` failed and the other three were skipped.

The conformance run used Sonobuoy's default non-disruptive profile, which checks that this cluster's configuration behaves as expected; it is not a substitute for the CNCF certification of Amazon EKS. The recovery check is not a CNTi test. It confirms that a worker restarted through the EC2 API comes back healthy, so that any later resilience failure can be attributed to the CNF rather than to the nodes.

NEXT STEPS

- **Run the remaining categories**. This run covered two of the suite's seven categories. The full `cnti-testsuite workload` set includes the Resilience category, whose node drain and chaos tests need several schedulable nodes, which the four-node cluster provides.
- **Test free5GC.** free5GC is the suite's production-representative reference CNF. Its user plane function needs the gtp5g kernel module and unsafe sysctls on the worker nodes, which on EKS requires node configuration beyond the default Amazon Linux 2023 image.
- **Publish the companion code.** The wrapper script, runner definition and CoreDNS configuration that reproduce this run will be made available. *(Location to be agreed between CNTi and AWS.)*
- **Contribute upstream.** Fixes and documentation found while running on a managed Kubernetes service go back to the CNTi Test Suite.

GET INVOLVED

Try the suite against your own CNF, starting with the CoreDNS example, and share results or questions with the CNTi community.

- CNTi Test Suite: **github.com/lfn-cnti/testsuite**
- Test documentation: **github.com/lfn-cnti/testsuite/blob/main/docs/TEST_DOCUMENTATION.md**
- Cloud Native Telecom Initiative: **lfnetworking.org/cloud-native-telecom-initiative**
- Amazon EKS Best Practices Guide: **docs.aws.amazon.com/eks/latest/best-practices**

About the author. Viyoma Sachdeva is a Principal Cloud Architect at AWS Professional Services who specializes in telecommunications network modernization and cloud-native transformations. She works with communications service providers on large-scale migration and modernization programs and is an active contributor to the Cloud Native Telecom Initiative (CNTi) at LF Networking.

About CNTi. This study was developed in collaboration with the Cloud Native Telecom Initiative (CNTi), an LF Networking project. CNTi maintains the open source, vendor-neutral CNTi Test Suite, which checks that telecom network functions follow Kubernetes and cloud native best practices and reports the result per test, with the evidence to back it. The Test Suite maintainers reviewed the CNTi content in this study. Learn more or contribute at **github.com/lfn-cnti/testsuite**.