# CNF Test Suite Intro

Thursday, July 1, 2021

Taylor Carpenter
taylor@vulk.coop

W. Watson
w.watson@vulk.coop

# About Taylor Carpenter

- **CNF WG Co-Chair**
- Co-Founder, Vulk Coop in Austin, TX
- 20+ years of devops + development experience
  - Telecom, Finance, Healthcare
- Open Source advocate, using Linux since 1994
- Helping CNCF with CI/CD and Cloud Native Telecom initiatives since 2017
- **Slack:** @taylor
- **GitHub:** @taylor

# About W. Watson

- **CNF Test Suite Maintainer**
- Co-Founder, Vulk Coop in Austin, TX
- Founder, Austin Software Cooperatives meetup
- Professionally developing software for 25+ years
  - Defense, medical, education, insurance
- Helping CNCF with CI/CD and Cloud Native Telecom initiatives since 2017
- **Slack:** @watson
- **GitHub:** @wavell

# Agenda

- CNCF's Telecom Initiatives
- CNF Test Suite Intro
- How to Use the CNF Test Suite
- Overview of Workload & Platform Tests
- What's Next?
- How to Contribute
- Feedback Requested

# CNCF Initiatives for Telecom

# Cloud Native Benefits for Telcos

By adopting cloud native technologies, Telcos are ensuring:

- **Better resource efficiency** to run the same number of services on fewer servers
- **Improved resiliency and availability** despite failures of individual CNFs, machines, or even data centers
- **Higher development velocity** with reduced risk
- **Interoperability improvements** to help with disaggregation and multi-vendor compatibility

# CNCF Initiatives for Telecom

**Cloud Native Principles and Best Practices**




# Telecom User Group (TUG)

A Telecom community user group

- **Meetings held monthly on the first Monday**
  - See https://github.com/cncf/telecom-user-group

**Goals:**

- Share ideas, ask questions, voice needs & concerns
- Review and discuss new/existing cloud native technologies
- Write up requirements, gap analysis, or similar documents
  - Created the Cloud Native Thinking for Telecommunications white paper
  - Contributed to the Cloud Native Networking papers

# Cloud Native Network Function Working Group (CNF WG)

A collaboration between Service Providers, CNF Developers and the Kubernetes community

- **Meetings held weekly on Mondays at 16:00 UTC**
  - See https://github.com/cncf/cnf-wg/

**Goals:**

- To identify cloud native best practices for running **CNFs on Kubernetes**, which CNF Developers & operators may adopt
- To determine which practices will allow **networking applications** to most effectively utilize Kubernetes capabilities and services

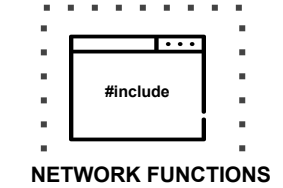
# CNF Testbed

An open source **K8s environment & toolchain** to create, deploy and test cloud native networking use cases

- See https://github.com/cncf/cnf-testbed

**Goals:**

- Test and review emerging cloud native technologies in the Telecom domain
- Provide **fully reproducible** use cases & examples
- Out-of-the box support for deployments to on-demand hardware from the bare metal hosting company, **Equinix Metal™**



# CNF Testsuite

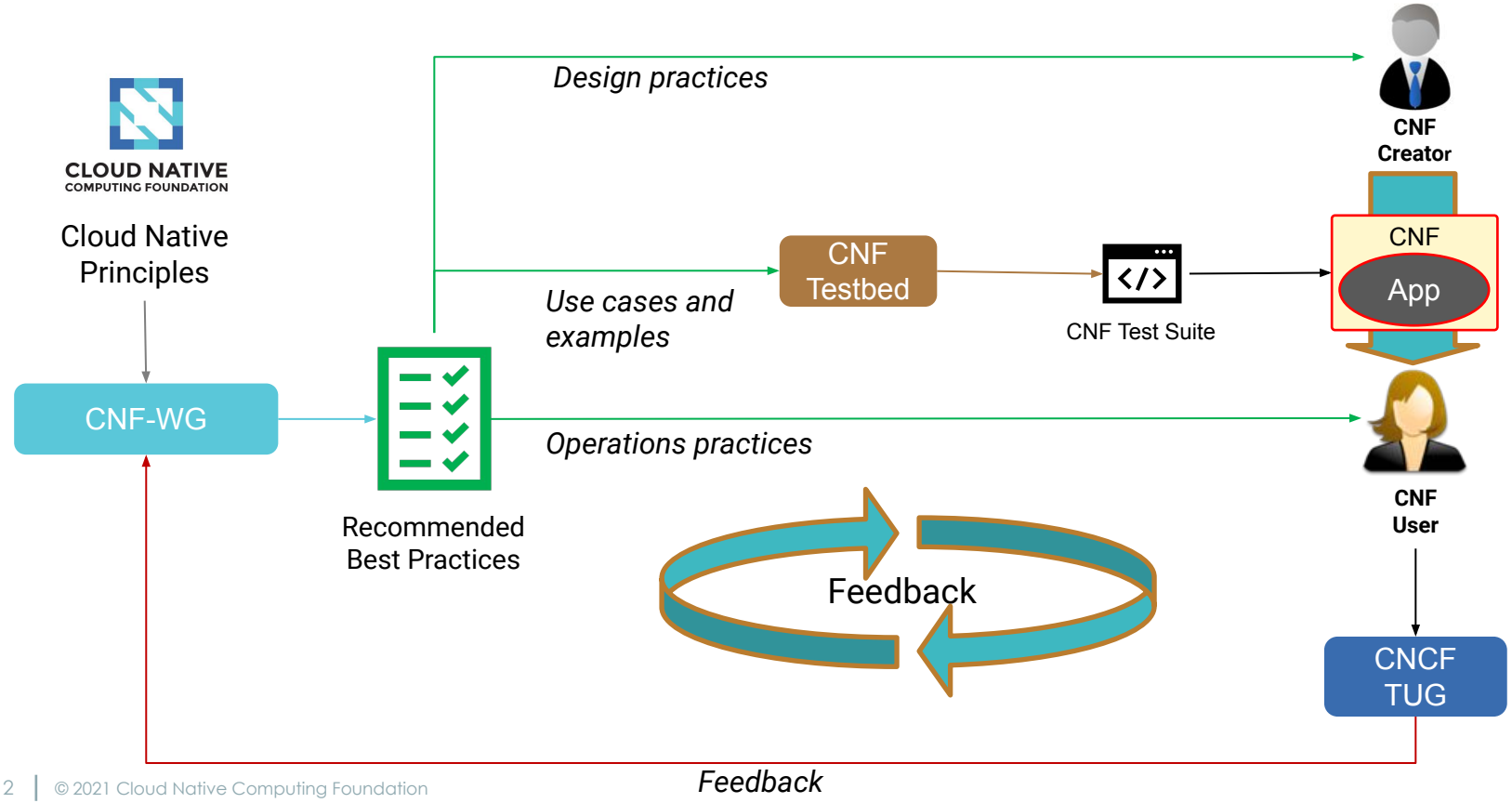
An open source **test suite** for CNF developers and network operators to evaluate how well a network application, aka **Cloud Native Network Function (CNF)**, follows cloud native principles and best practices.

- See https://github.com/cncf/cnf-testsuite

**Goals:**

- Easy to install and use
- Fast feedback to end user
- Out-of-the box support for certified K8s environments

# How They Relate




 | 

# CNF Test Suite

CLOUD NATIVE
COMPUTING FOUNDATION

# What is the CNF Test Suite?

An open source **test suite** for CNF developers and network operators to evaluate how well a network application, aka **Cloud Native Network Function (CNF)**, follows cloud native principles and best practices.

This test suite initiative works closely with the CNF Working Group which identifies best practices.

# Why Use Cloud Native Best Practices?

Implementing and running applications in a cloud native manner will enable you to more fully benefit from the advantages of cloud native infrastructure.

- **Shared experience:** build upon the work of the community
- **Interoperability**: Standardization for communication between applications and the cloud platforms
- **Predictability**: Your application acts in a predictable manner when running on cloud native infrastructure like Kubernetes. Unexpected behavior should be rare because application specific issues are weeded out during the best practice testing.

 | 

# Why Use the CNF Test Suite?

Designed to help developers and operation teams to adopt and improve cloud native practices

- **Faster feedback loop**
- Integrated with your existing **CI/CD pipelines**
- Aligned with upstream **CNCF ecosystem**




# Features

- Is **self-contained** with minimal requirements and necessary configuration
- Provides suggestions on how to improve a network application to better **follow cloud native best practices**
- Supports self-hosted and protected image repositories
- Supports **air-gapped environments**
- Gives **fair assessment** with a flexible scoring system
  - Tests fail gracefully
  - Tests are skipped when prerequisites are not met




# Implementation Overview

- The test framework and tests are written in the human-readable, compiled language, **Crystal**. Common capabilities like dependencies between tests and categories are supported.

- The CNF Test Suite leverages **upstream tools** where possible such as LitmusChaos, Chaos Mesh, kubectl, dockerd, and Helm linter for testing CNFs. The upstream tool installation, configuration, and versioning has been made repeatable.

- Setup of **vanilla upstream K8s** on Equinix Metal™ is done with the CNF Testbed platform tool chain, which includes k8s-infra and Kubespray. Testing also occurs in kind clusters daily.

# Contributors


@taylor


@wavell


@nupejosh

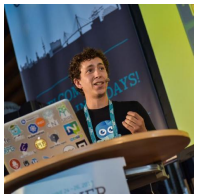

@lixuna


@denver williams


@williscool


@agentpoyo

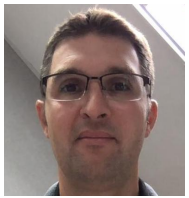

@HashNuke


@xmulligan


@michaels pedersen


@petorre


@electro cucaracha


@sishbi


@clementnuss


@uditgaurav


@saksham gurbhele

https://github.com/cncf/cnf-testsuite/graphs/contributors

# Demo, Installation, and Setup

# Live Demo

- Example CNF based on CoreDNS (a CNCF graduated project)
  - Modified to demonstrate **passed**, **skipped** and **failed** test results
- Run Workload Tests
  - Duration: 5-6 minutes

# Installation, Setup, and Running

To get the CNF Test Suite up and running, see the Installation Guide.

## To give it a try immediately, you can use these quick install steps

**Prereqs:** Kubernetes cluster, kubectl, curl, helm 3.1.1 or greater on your system already

1. Install the latest test suite binary: `source <(curl https://raw.githubusercontent.com/cncf/cnf-testsuite/main/curl_install.sh)`
2. Run `setup` to prepare the CNF Test Suite: `cnf-testsuite setup`
3. Download an example CNF configuration to try: `curl -o cnf-testsuite.yml https://raw.githubusercontent.com/cncf/cnf-testsuite/main/example-cnfs/coredns/cnf-testsuite.yml`
4. Initialize the test suite for using the CNF: `cnf-testsuite cnf_setup cnf-config=./cnf-testsuite.yml`
5. Run all of application/workload tests: `cnf-testsuite workload`

https://github.com/cncf/cnf-testsuite/blob/main/README.md#installation-and-usage




# Testing Feedback

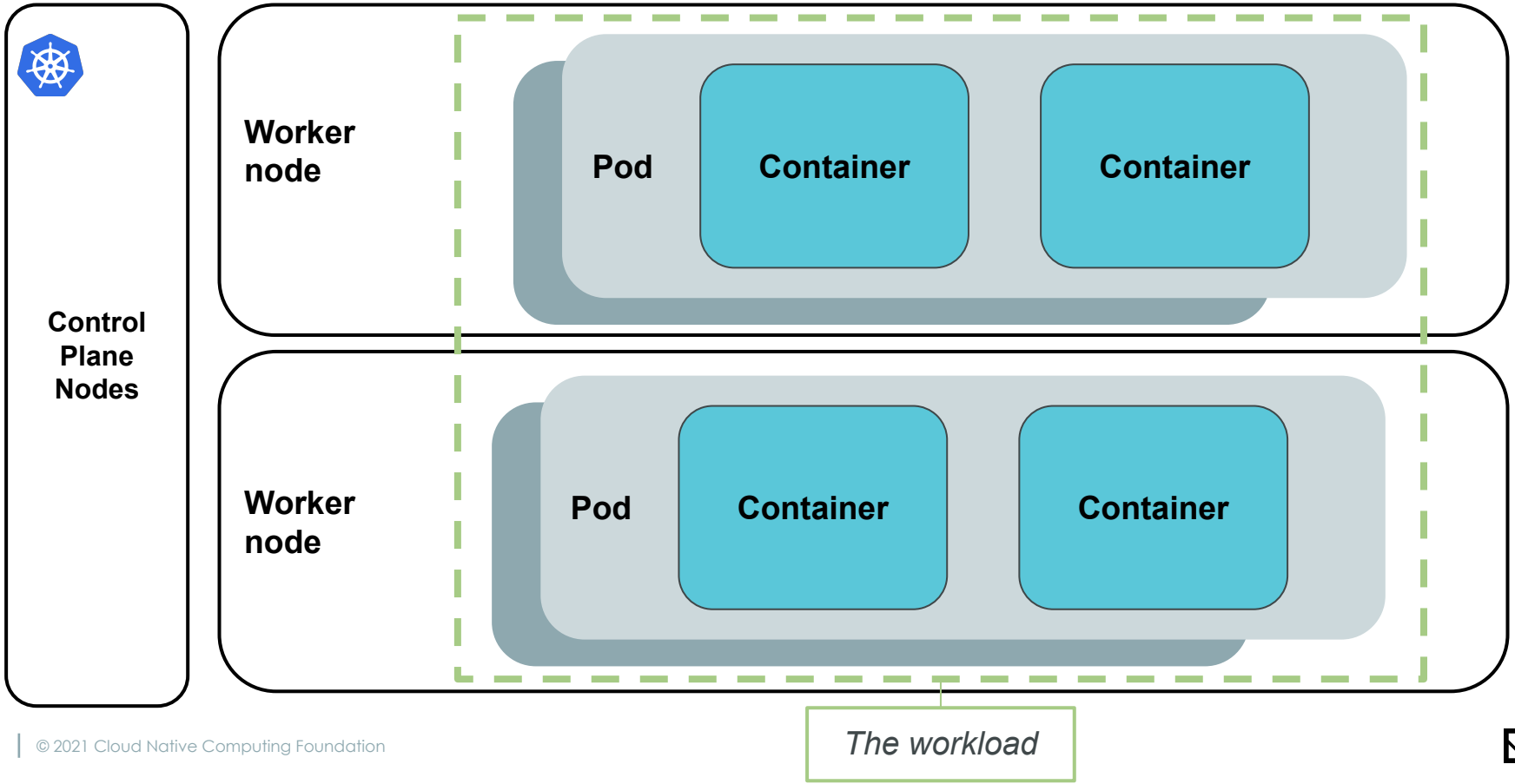
```
✓  PASSED: No privileged containers 🔓🔧
Security final score: 5 of 5

✓  PASSED: Replicas increased to 3 📦📈
✓  PASSED: Replicas decreased to 1 📦📉
Scalability final score: 20 of 35

✓  PASSED: No IP addresses found
✓  PASSED: Helm liveness probe found ⎈🫁
✓  PASSED: Helm readiness probe found ⎈🫁
✓  PASSED: NodePort is not used
✓  PASSED: No hard-coded IP addresses found in the runtime K8s configuration
✓  PASSED: CNF Rollback Passed
No Secret Volumes or Container secretKeyRefs found for resource: {kind: "Deployment", name: "coredns-coredns"}
⏭  SKIPPED: Secrets not used 🫁

To address this issue please see the USAGE.md documentation

✗  FAILED: Found mutable configmap(s) ⚖️
✓  PASSED: CNF for Rolling Update Passed
✓  PASSED: CNF for Rolling Downgrade Passed
✓  PASSED: CNF for Rolling Version Change Passed
Configuration Lifecycle final score: 40 of 46

✓  PASSED (by default): No install script provided
Successfully created directories for cnf-testsuite
✓  PASSED: Helm Chart exported_chart Lint Passed ⎈📝☑️
✗  FAILED: Published Helm Chart Not Found ⎈📦🌐
SKIPPED: Helm Deploy
Installability final score: 9 of 20

✓  PASSED: Image size is good 🦐 ⚖️👀
✓  PASSED: CNF had a reasonable startup time 🚀
Microservice final score: 10 of 10

✓  PASSED: pod_network_latency chaos test passed 🔪💀♻️
✓  PASSED: Application pod is healthy after high CPU consumption 📦💻🐷📈
✓  PASSED: Replicas available match desired count after container kill test 🔪💀♻️
```

# Overview of Instructions & Usage

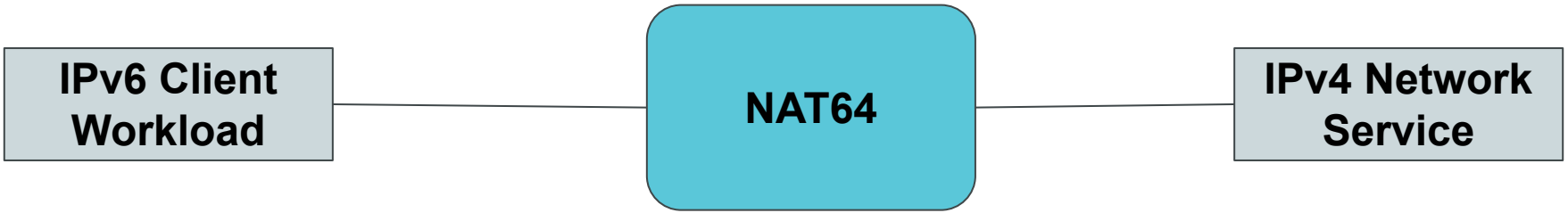
- Installation instructions are published on the GitHub repo
  - CNF Developer step-by-step walk through
  - Test developer / source install instructions
- Usage documentation
- Prerequisites information
- Configuration instructions for your own CNF
- Example CNFs

# CNF Test Suite Workload Tests

# What is a Workload?



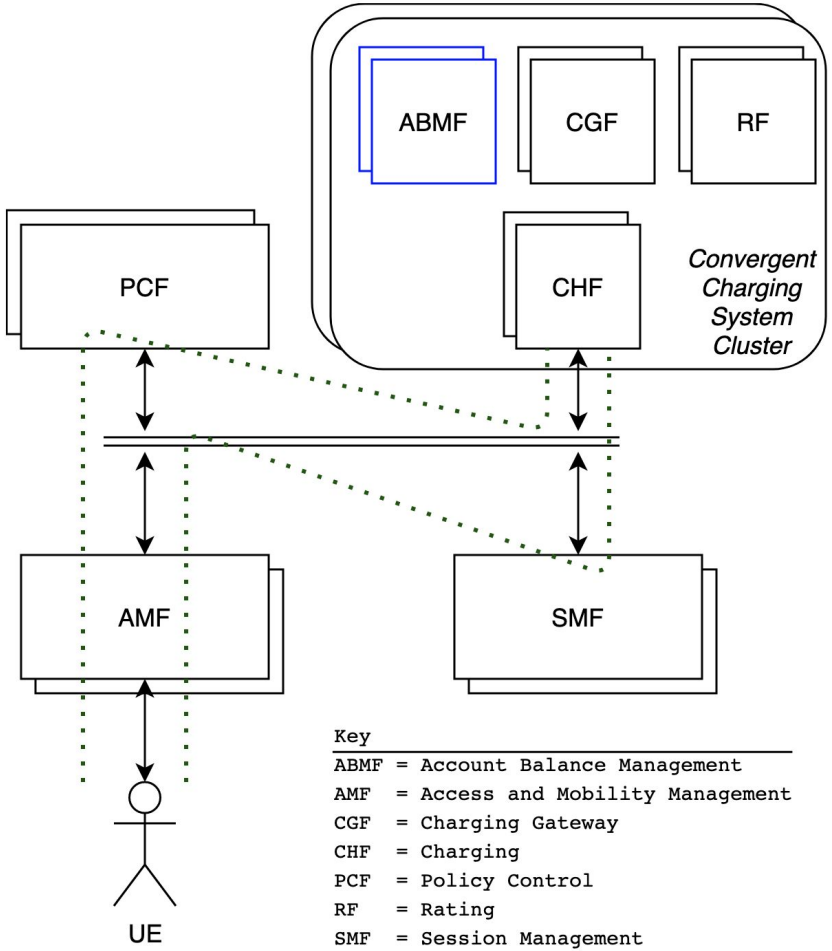
# Simple to...




 | 

# Simple to...

# Complex




Key

ABMF = Account Balance Management
AMF = Access and Mobility Management
CGF = Charging Gateway
CHF = Charging
PCF = Policy Control
RF = Rating
SMF = Session Management
UE = User Equipment

# Test Categories

| Installable | Configuration | Microservice | Compatibility | State | Security | Scalability | Observability | Resilience |
|---|---|---|---|---|---|---|---|---|
| Published Helm chart | Liveness | Startup time | N/A | Volume HostPath Not Found | Privileged mode | Increase capacity | Traffic to Prom-etheus | CNF back up when container killed/dies |
| Valid Helm chart | Readiness | Image Size | | No Local Volume Config-uration | | Decrease capacity | | LitmusChaos Pod Network Loss |
| Helm Deploy | Rolling version upgrade/ downgrade | | | | | ClusterAPI enabled | | LitmusChaos Pod Network Latency |
| Install Script Helm | No nodePort | | | | | | | LitmusChaos Pod CPU Hog |
| Rolling Update | Secrets Used | | | | | | | LitmusChaos Disk Fill |
| | Static: Hardcoded IPs in source code | | | | | | | LitmusChaos Pod Delete |
| | Immutable Config Maps | | | | | | | |




# CNF Test Suite Platform Tests

# Current Platform Testing

- **Kubernetes-based** platforms
- OCI Compliance
- Worker Node Failure (*destructive*)
  - *Note: reboots node via privileged container*
- Prometheus Observability

 | 

# Running Platform Tests

For **platform testing,** only 3 steps are needed, including the common install and setup steps:

1. Install the latest test suite binary: `source <(curl -s https://raw.githubusercontent.com/cncf/cnf-testsuite/main/curl_install.sh)`
2. Run `setup` to prepare the cnf-testsuite suite: `cnf-testsuite setup`
3. **Run all of platform tests: `cnf-testsuite platform`**

 | 

# What's Next?

- Add cloud native best practice **suggestions** to test results
- Build more **resilience** tests using LitmusChaos experiments
- Create **observability** tests to check for cloud native monitoring
- Create more **state** tests to check cloud native data handling
- Validate best practices as defined by **CNF WG**
- Split libraries out into **multi-repositories** under a single organization
- Onboard maintainers and end users

# CNF Test Suite Results

# Live Demo

- Review workload **test results**
- Example CNF based on CoreDNS (a CNCF graduated project)
  - Modified to demonstrate **passed**, **skipped** and **failed** test results




# Test Results

- Results for tests are displayed on the console and saved to disk

```
✔  PASSED (by default): No install script provided
Successfully created directories for cnf-testsuite
✔  PASSED: Helm Chart exported_chart Lint Passed ⎈📝☑
✖  FAILED: Published Helm Chart Not Found ⎈📦🌐
SKIPPED: Helm Deploy
Installability final score: 9 of 20

✔  PASSED: Image size is good 🐜 ⚖👀
✔  PASSED: CNF had a reasonable startup time 🚀
Microservice final score: 10 of 10

✔  PASSED: pod_network_latency chaos test passed 🗡💀♻
✔  PASSED: Application pod is healthy after high CPU consumption 📦💻🐷📈
✔  PASSED: Replicas available match desired count after container kill test 🗡💀♻
✔  PASSED: disk_fill chaos test passed 🗡💀♻
Resilience final score: 10 of 10

Final workload score: 109 of 136
CNFManager::Points::Results.have been saved to results/cnf-testsuite-results-20210526-182222-063.yml
```

# CNF Test Suite Summary

- **Faster feedback loop**
  - Runs workload tests in under 10 minutes
- Integrated with your existing **CI/CD pipelines**
  - Has been tested with GitHub Actions, Travis CI & Jenkins CI (ex. OPNFV Functest integration)
- Aligned with upstream **CNCF ecosystem**
  - Checks compatibility with CNCF projects, like K8s Conformance, Helm & LitmusChaos

# How to Contribute

# Ready to Get Started?

- **Run the CNF Test Suite**
  - CNF Developer Install and Usage guide
  - Provide **feedback** via GitHub or slack
- **Contributions welcome**
  - Suggest enhancements
  - Report bugs
  - Request/add new tests
  - Request/add CNFs to be validated
  - Documentation updates
  - Watch, star and/or fork the CNF Test Suite GitHub repo

 | 

# Feedback Requested

- How would you like to utilize the CNF Test Suite?
- In what **format** would you like to receive the test results (ie. JUnit XML, TAP protocol, etc)?
- What are some **use cases** that are important to your business?
- What are some of the **challenges** experienced while moving towards cloud native and using a Kubernetes based environment?
- Do you have any specific **hardware resource management** challenges?
- Are you seeing any **resilience issues** for the workloads you are running?
- How many **requests** or **packets per second** does your sample CNF get?

 | 

# Join the Conversation

## CNF Working Group Meeting (Mondays at 16:00 - 17:00 UTC)

- CNF WG Meeting Details
- Mailing List: https://lists.cncf.io/g/cnf-wg

## CNF Test Suite & Testbed Contributor Meeting (Thursdays at 14:15 - 15:00 UTC)

- CNF Test Suite Meeting Details
- Mailing List: https://lists.cncf.io/g/cnf-test-suite

## CNCF Slack Channels

- slack.cncf.io
  - #cnf-testsuite-dev
  - #cnf-testbed-dev
  - #cnf-wg




# Contact

**Taylor Carpenter, Vulk Coop**
taylor@vulk.coop
GitHub: @taylor
Slack: @taylor
Schedule a demo

**Bill Mulligan, CNCF**
bmulligan@linuxfoundation.org
GitHub: @xmulligan
Slack: @Bill Mulligan
Meet with Bill

 | 

# Thank You!

CLOUD NATIVE
COMPUTING FOUNDATION

# Appendix

# CNF Testsuite

An open source **test suite** for CNF developers and network operators to evaluate how well a network application, aka **Cloud Native Network Function (CNF)**, follows cloud native principles and best practices.

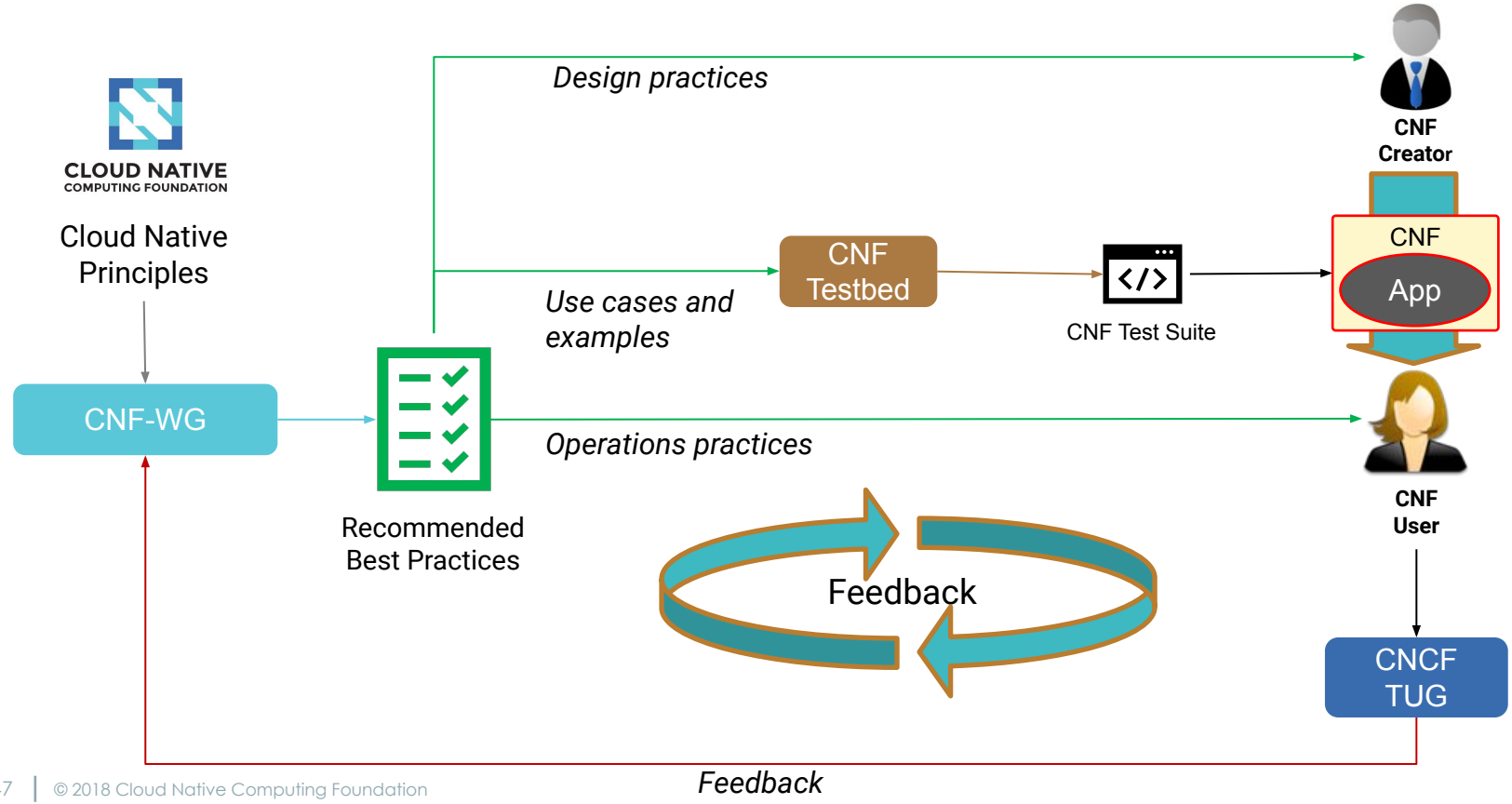
- See https://github.com/cncf/cnf-testsuite

**Goals:**

- Easy to install and use
- Fast feedback to end user
- Out-of-the box support for certified K8s environments




# How They Relate




 | 

# CNF Test Suite Results + OPNFV functest

Initial integration of the CNF Test Suite with OPNFV Functest is complete and running CNF workload tests

| TEST CASE | PROJECT | DURATION | RESULT |
|---|---|---|---|
| cnf_conformance | functest | 02:06 | PASS |

https://gerrit.opnfv.org/gerrit/c/functest-kubernetes/+/70914

# Cloud Native Network Function (CNF) Definition

A cloud native network function (CNF) is a cloud native application that implements or facilitates network functionality.

A cloud native network function consists of one or more microservices, and has been developed using Cloud Native Principles including **immutable infrastructure**, **declarative APIs**, and a "**repeatable deployment process**."

https://networking.cloud-native-principles.org/




# Problem Statement

With the current User Plane for an Evolved Packet Core, Telco engineers are not satisfied with:

- Complexity in network function implementation and life-cycle management
- Handling failures
  - Application and platform outages still happen
  - Lack of auto-healing
- How long it takes to promote changes to production

 | 

# Solution

Creating a Cloud Native User Plane for an Evolved Packet Core (EPC):

- Building cloud native versions of the Serving Gateways (S-GW) and Packet Data Network (PDN) Gateways (P-GW)
- Using Kubernetes platform and add-ons to provide some of the services and functionality traditionally implemented in the network functions used in a EPC user plane

# Advantages

A Cloud Native User Plane for an Evolved Packet Core will:

- Reduce complexity of NF implementation
- Reduce complexity of life cycle management
- Increase resiliency
- Provide horizontal auto-scaling
- Increase delivery velocity to production

# CNF Examples

# Current CNF Examples

- Example-cnfs
  - **CoreDNS**
  - **Envoy**
  - **IP-forwarder**

# Future CNF Examples to Test

- EXAMPLE-CNFS.md
  - **free5GC (https://www.free5gc.org/)**
  - **vBNG from ONAP BBS use-case**
  - **Go-GTP based example**
  - **Envoy with WASM**
  - **5G Packet Core**
  - **Scaling P-GW**

Service chains, use cases, and CNFs with dependent services




# Q&A