CNFs vs. VNFs

Dan Kohn
Executive Director, CNCF
Cloud Native Computing Foundation

- Non-profit, part of the Linux Foundation; founded Dec 2015

<table>
<thead>
<tr>
<th>Graduated</th>
<th>Incubating</th>
<th>Sandbox</th>
</tr>
</thead>
<tbody>
<tr>
<td>kubernetes</td>
<td>OPENTRACING</td>
<td>ROOK</td>
</tr>
<tr>
<td>Orchestration</td>
<td>Distributed Tracing API</td>
<td>Storage</td>
</tr>
<tr>
<td>Prometheus</td>
<td>fluentd</td>
<td>spiffe</td>
</tr>
<tr>
<td>Monitoring</td>
<td>GRPC</td>
<td>Identity Spec</td>
</tr>
<tr>
<td></td>
<td>Remote Procedure Call</td>
<td>SPIRE Identity</td>
</tr>
<tr>
<td></td>
<td>Container</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Runtime</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Platinum members:

- Alibaba Cloud
- AWS
- Azure
- Cisco
- Dell Technologies
- Docker
- Fujitsu
- Google Cloud
- IBM Cloud
- Intel
- JD.COM
- Mesosphere
- Oracle
- Pivotal
- Red Hat
- Samsung
- SAP
- VMware
TODAY THE LINUX FOUNDATION IS MUCH MORE THAN LINUX

Security
We are helping global privacy and security through a program to encrypt the entire internet.

Networking
We are creating ecosystems around networking to improve agility in the evolving software-defined datacenter.

Cloud
We are creating a portability layer for the cloud, driving de facto standards and developing the orchestration layer for all clouds.

Automotive
We are creating the platform for infotainment in the auto industry that can be expanded into instrument clusters and telematics systems.

Blockchain
We are creating a permanent, secure distributed ledger that makes it easier to create cost-efficient, decentralized business networks.

Web
We are providing the application development framework for next generation web, mobile, serverless, and IoT applications.

We are regularly adding projects; for the most up-to-date listing of all projects visit tlfprojects.org
KubeCon + CloudNativeCon

• China
  – Shanghai: November 14-15, 2018
  – Sponsorships open
• North America
  – Seattle: December 11 - 13, 2018
  – Sponsorships open
• Europe
  – Barcelona: May 21 - 23, 2019
Network Architecture Evolution

› 1.0: Separate physical boxes for each component (e.g., routers, switches, firewalls)
› 2.0: Physical boxes converted to virtual machines called Virtual Network Functions (VNFs) running on VMware or OpenStack
› 3.0: Cloud-native Network Functions (CNFs) running on Kubernetes on public, private, or hybrid clouds
Network Architecture 1.0
Network Architecture Evolution

› 1.0: Separate physical boxes for each component (e.g., routers, switches, firewalls)
› 2.0: Physical boxes converted to virtual machines called Virtual Network Functions (VNFs) running on VMware or OpenStack
Network Architecture 2.0
Network Architecture Evolution

› 1.0: Separate physical boxes for each component (e.g., routers, switches, firewalls)
› 2.0: Physical boxes converted to virtual machines called Virtual Network Functions (VNFs) running on VMware or OpenStack
› 3.0: Cloud-native Network Functions (CNFs) running on Kubernetes on public, private, or hybrid clouds
Network Architecture 3.0
(hardware is the same as 2.0)
Evolving from VNFs to CNFs

- **Past**: VNFs run on OpenStack or VMWare, or Azure or Rackspace. ONAP Orchestrator runs on OpenStack or VMWare, or Azure or Rackspace.
- **Present**: VNFs run on OpenStack or Kubernetes, or Bare Metal. CNFs run on Kubernetes or any cloud. ONAP Orchestrator runs on Kubernetes or any cloud.
- **Future**: CNFs run on Kubernetes or any cloud. VNFs run on KubeVirt/Virtlet or Kubernetes. ONAP Orchestrator runs on Kubernetes or any cloud.

- ONAP Amsterdam (Past) runs on OpenStack, VMware, Azure or Rackspace.
- ONAP Casablanca (Present) runs on Kubernetes and so works on any public, private or hybrid cloud.
- Virtual Network Functions (VNFs) are virtual machines that run on OpenStack or VMWare, or can be run on K8s via KubeVirt or Virtlet.
Major Benefits

1. Cost savings (with public, private, and hybrid clouds)
2. Development velocity
3. Resiliency (to failures of individual CNFs, machines, and even data centers)
The challenge of transitioning VNFs to CNFs

› Moving from network functionality from physical hardware to encapsulating the software in a virtual machine (P2V) is generally easier than containerizing the software (P2C or V2C)

› Many network function virtualization VMs rely on kernel hacks or otherwise do not restrict themselves to just the stable Linux kernel userspace ABI
  › They also often need to use DPDK or SR-IOV to achieve sufficient performance

› Containers provide nearly direct access to the hardware with little or no virtualization overhead
  › But they expect containerized applications to use the stable userspace Linux kernel ABI, not to bypass it
Areas for More Discussion

› The strength of no longer being locked into specific OSs
  › Any version of Linux >3.10 is acceptable
› Multi-interface pods vs. Network Service Mesh
› Complete parity for IPv6 functionality and dual-stack support in K8s
› Security, and specifically recommendations from Google and Jess that come into play when hosting untrusted, user-provided code
  › Possible use of isolation layers such as gVisor or Kata
› Scheduling container workloads with network-related hardware constraints (similar to what’s been done for GPUs)
  › Network-specific functionality like QOS
Demo Plans Underway

› VNFs vs. CNFs
  › Working on a demo of boot-time and throughput of VNFs on OpenStack vs. CNFs on Kubernetes, where the networking code and underlying hardware is identical
  › Will deliver open source installers and Helm charts

› Cloud-native Customer Premises Equipment (CCPE) Project
  › Modify the ONAP vCPE use case and VNF deployment to show VNF vs. CNF deployments of chained network functions
Roll-Out Plans

› **Open Source Summit NA**, Vancouver, August 28: Joint workshop by CNCF executive director Dan Kohn and LF Networking head Arpit Joshipura on Cloud-native Network Functions

› **Open Network Summit Europe**, Amsterdam, September 25: Marketing launch

› **KubeCon + CloudNativeCon NA**, Seattle, December 11: Planned demo

› **Mobile World Congress**, Barcelona, February 25: Major roll-out

› Ongoing close collaboration with LF Networking and specific carriers providing feedback (AT&T, Bell Canada, Vodafone, etc.)
The Networking aspects of Cloud Native

Arpit Joshipura
GM Networking
The Linux Foundation
Industry Direction: Any Cloud + Portable Apps in Containers

› Utilize the best of Cloud with the best of Telecom Networks
  › Promise of Containers – allow for any cloud portability
  › Promise of Network – full network automation & zero touch services

› Telecom Network Transformation require a hybrid strategy
  › Migration of VM to containers step by step approach to VNF/Workloads
  › VM and Container Interworking in a Multi-VIM environment
Two leading de-facto platforms – Networking & Cloud

- Network Automation & Orchestration Platform
- ONAP has a multi-VIM strategy (Openstack, Vmware, Azure,..)
- Project within ONAP – OOM looking at Containers

- Cloud Automation & Orchestration Platform
- Project within CNCF looking at ONAP – Cross Cloud CI

Open Source projects at LF can bring the best of both worlds to the Telecom Industry
Sustainable Innovation: Open Source Networking
Creating De-Facto Platforms to Enable Next Generation Solutions in Telecom, Enterprise & Cloud

Carrier Services
Cloud Services
Enterprise Services
10/10 Top Vendors Active

Network Automation/Zero Touch
New Services in Minutes
$576M Shared Innovation
SDO+OSS Harmonization
New Services 5G/IOT/Edge/AI
LF 9/10 Most important projects

De-Facto Platform for ~70% Global Sub
LF Networking Vision: Automating Cloud, Network, & IOT Services

Service

Cloud Services  Residential Services  Enterprise Services  IOT Services

Software & Automation

Cloud Automation

IOT Automation

Infrastructures

Enterprise
Software Defined Data Centers (SDDC)

Cloud Automation

(ONAP, OPNFV, ODL, FD.io, SNAS, PNDA)

Public/Hybrid
Cloud Service Providers
Cloud Hosting
Private Cloud Providers
Web Service Providers

Service Providers
MSO/CableCo

Data Centers  Carrier Network  Cloud Network
Bringing It All Together Core to Edge – LF Open Source Network + Disaggregation + Edge + IOT + AI + Cloud + Blockchain

Standards for Edge

Ref Implementation

IoT, Gateway & Cloud Ref Arch

Other Edge Activities
Open Source Networking Landscape
Linux Foundation hosts 9/10 Top projects

Service Levels
- Application Layer / App Server
- Network Data Analytics
- Orchestration, Management, Policy
- Cloud & Virtual Management
- Network Control
- Network Operating Systems
- IO Abstraction & Data Path
- Disaggregated Hardware

Product, Services & Workloads

Automation of Network + Infrastructure + Cloud + Apps + IOT
Linux Foundation Path to Open Source Harmonization 2.0

Key Drivers of Each Layer

- AI & Marketplace/By Vertical X Projects
- Core to Edge Zero Touch Automation
- VM to Container Migration, Portability
- Integrated Edge Stack – Zero Touch
- Include OpenStack, Azure, RS, VMware…
- Apps, Location and Service Portability
- IIOT Framework For Core Services

Layer: Edge
- AcumosAI / Deep Learning / Blockchain
- LFN / ONAP+ODL+OPNFV+FD.io
- CNCF / Kubernetes
- Akraino Edge Stack
- Hybrid Orchestration/ VIMs
- Any Cloud (Public, Hybrid, Service Provider Core, Edge…)
- EdgeX Foundry

Layer: Carrier Cloud & Enterprise

Analytics/AI/Blockchain
- Automation, Control & Orchestration

Infrastructure

Devices/IoT
- Services
The deep dive – VNFs on ONAP & Cloud Native journey
ONAP Beijing Architecture

VNF Requirements Model (Utilities)

VNF Validation Program
A Day in the Life of an ONAP Service

1. VNF
   Vendor provided VNF (cloud-hosted, optimized or native)

2. Design
   Design/test teams onboard VNFs

3. Vendor packages VNF as per ONAP requirements; can use VNF SDK

4. Designers create products, services, recipes

5. OSS/BSS system triggers service deployment

6. Service lifecycle management

7. Constant data collection, analytics, event monitoring; S3P

Credit: Aarna Networks, ONAP Training course
Kubernetes Gap Analysis & Transition plans

Top 3 Areas of Investigation

1. Support for VNFs/Apps from different vendors (Ref guest OS and VNF ecosystem alignment)
2. Security – Access rights and privileges, known rules for admins etc
3. Network Specific Requirements – focus on performance, scale and capabilities
   › Enabling scheduling container workloads with network-related hardware constraints (similar to what’s been done for GPUs)
   › Multi-homed containers (Multis, a CNI plugin is working on this)
   › Functional parity when deployed with IPv6
   › Network-specific functionality like QOS
   › multiple vNIC for a given container, which is currently not supported.

Transition plan

› Demo of Any app, Any cloud with ONAP running on Kubernetes @ ONS opening keynote
› Kubernetes as the Virtual Infrastructure Manager (VIM) for running the ONAP Orchestrator
› Consider Kubernetes as the ONAP Application Controller (AppC)
› As VNFs can be containerized, do so and manage them with Kubernetes – prioritize use cases (eg CDN, DNS)
Evolving from VNFs to CNFs

- ONAP Amsterdam (Past) runs on OpenStack, VMware, Azure or Rackspace
- ONAP Casablanca (Present) runs on Kubernetes and so works on any public, private or hybrid cloud
- Virtual Network Functions (VNFs) are virtual machines that run on OpenStack or VMWare, or can be run on K8s via KubeVirt or Virtlet
Today’s Agenda

BACKGROUND AND VISION
1:30 Introduction to VNFs and CNFs & Cross-cloud Dan Kohn
2:00 Networking & Telecom Automation: VNF to CNF journey Arpit Joshipura

REQUIREMENTS
2:15 Cloud Native lessons and requirement: A view from end user - Telus, Sanah Tariq
2:30 (Dan/Arpit facilitate) Why Telecom and Cloud Native technologies are coming together – discuss challenges and requirements

BREAK

PROJECTS AND ROADMAP
3:30 Overview of projects solving the migration Roadmap to Cloud Native
3:50 Network Service Mesh (VPP/Ligato) Ed Warnicke
4:10 Cross-Cloud CI working group Taylor
4:30 Wrap up and How to get involved
Envoy & Istio overview

Ihor Dvoretskyi, @idvoretskyi,
Developer Advocate, CNCF
• **Inception** May ‘15
• **Open sourced** September ‘16: [https://github.com/envoyproxy/envoy](https://github.com/envoyproxy/envoy)
• **Joined CNCF** September’17: [https://www.envoyproxy.io/](https://www.envoyproxy.io/)
• **Users/contributors** (partial list): Lyft, Google, IBM, Verizon, Apple, Microsoft, Pivotal, Red Hat, EBay, Stripe, VSCO, Tencent QQ, Twilio, Yelp … and more all the time.
• **Integrations** (partial list): Istio (Google/IBM), Nomad (Verizon), Tigera, Covalent, Turbine Labs, Datawire … more on the way.
• **Lyft deployment**: 10s of thousands of hosts, 100s of services, 3M + mesh RPS.
State of Service-Oriented Architecture networking

- **Protocols** (HTTP/1, HTTP/2, gRPC, databases, caching, etc.).
- **Infrastructures** (IaaS, CaaS, on premise, etc.).
- Intermediate **load balancers** (AWS ELB, F5, etc.).
- **Observability** output (stats, tracing, and logging).
- Implementations (often partial) of **retry**, **circuit breaking**, **rate limiting**, **timeouts**, and other distributed systems best practices.
- **Authentication** and **Authorization**.
- Per language **libraries** for service calls.
Envoy overview

- **Out of process architecture**: Let’s do a lot of really hard stuff in one place and allow application developers to focus on business logic.
- **Modern C++ code base**: Fast and productive.
- **L3/L4 filter architecture**: A byte proxy at its core. Can be used for things other than HTTP (e.g., MongoDB, redis, stunnel replacement, TCP rate limiter, etc.).
- **HTTP L7 filter architecture**: Make it easy to plug in different functionality.
- **HTTP/2 first!** (Including gRPC and a nifty gRPC HTTP/1.1 bridge).
- **Service discovery** and **active/passive health checking**.
- **Advanced load balancing**: Retry, timeouts, circuit breaking, rate limiting, shadowing, outlier detection, etc.
- Best in class **observability**: stats, logging, and tracing.
- **Edge proxy**: routing and TLS.
Connect, manage and secure microservices
Separation of concerns: developer and operations
For both containerized and non-containerized workloads
Leverage great functionality in Envoy, adding pluggable management and control planes
Intelligent routing, load balancing, metrics collection, policy enforcement, end-to-end authentication
Istio 1.0 announced in July’2018
Istio components

- **Mixer** - Central component that is leveraged by the proxies and microservices to enforce policies such as authorization, rate limits, quotas, authentication, request tracing and telemetry collection.
- **Pilot** - A component responsible for configuring the proxies at runtime.
- **Citadel** - A centralized component responsible for certificate issuance and rotation.
- **Node Agent** - A per-node component responsible for certificate issuance and rotation.
Envoy - Sidecar proxies per microservice to handle ingress/egress traffic between services in the cluster and from a service to external services

- Providing a rich set of functions like:
  - discovery
  - L7 routing
  - circuit breakers
  - policy enforcement
  - telemetry recording/reporting functions.
More details

• Envoy:
  – https://www.envoyproxy.io/
  – https://github.com/envoyproxy/envoy

• Istio:
  – https://istio.io/
  – https://github.com/istio/istio
Extra
Envoy overview

Service Cluster

Service

Envoy

External Services

Discovery

Service Cluster

Service

Envoy

HTTP/2

REST / gRPC

External Services

Discovery
Architectural overview

Traffic is transparently intercepted and proxied. Application is unaware of Envoy’s presence.
Bookinfo application sample
Cross-Cloud CI Overview

Cloud-native Network Functions Seminar
August 28, 2018
Presented by:

Taylor Carpenter, Vulk Coop
taylor@vulk.coop
Cross-Cloud CI Project Overview

Why?
Cross-Cloud CI Project Overview

Why? CNCF ecosystem is growing rapidly with new projects and cloud providers!
Cross-Cloud CI Project Overview

Why?

• The CNCF ecosystem is large, diverse and continues to grow. CNCF would like to ensure cross-project interoperability and cross-cloud deployments of all cloud native technologies and show the daily status of builds and deployments on a status dashboard.
Cross-Cloud CI Project Overview

Why?
• The CNCF ecosystem is large, diverse and continues to grow. CNCF would like to ensure cross-project interoperability and cross-cloud deployments of all cloud native technologies and show the daily status of builds and deployments on a status dashboard.

What?
• Cross-cloud testing system
• Status repository server
• Status dashboard
Build and provision CNCF projects

Graduated

- Kubernetes: Orchestration
- Prometheus: Monitoring

Incubating

- OpenTracing: Distributed Tracing API
- Fluentd: Logging
- gRPC: Remote Procedure Call
- containerd: Container Runtime
- rkt: Container Runtime
- CNI: Networking API
- Envoy: Service Mesh
- Jaeger: Distributed Tracing
- Notary: Security
- TUF: Software Update Spec
- Vitess: Storage
- CoreDNS: Service Discovery
- NATS: Messaging
- Linkerd: Service Mesh
- Helm: Package Management

Sandbox

- Rook: Storage
- SPIFFE: Identity Spec
- SPIRE: Identity
- Open Policy Agent: Policy
- CloudEvents: Serverless
- Telepresence: Tooling
- Harbor Registry
- OpenMetrics: Metrics Spec
- TiKV: Key/Value Store

© 2018 Cloud Native Computing Foundation
Project CI artifacts and non-CNCF projects

Implemented

ONAP
Open Network Automation Platform
Deploy to public/bare metal/private clouds

Implemented

- AWS
- Azure
- Google Cloud
- IBM Cloud
- openstack
- packet
- VMware

In Progress

- Oracle
<table>
<thead>
<tr>
<th>Project</th>
<th>Build</th>
<th>Release</th>
<th>Deployments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status</td>
<td>Stable</td>
<td>AWS</td>
</tr>
<tr>
<td>Kubernetes Orchestratior</td>
<td>SUCCESS</td>
<td>v1.11.2</td>
<td>Running</td>
</tr>
<tr>
<td>Prometheus Monitoring</td>
<td>SUCCESS</td>
<td>0.3a6d89b</td>
<td>Running</td>
</tr>
<tr>
<td>CoreDNS Service Discovery</td>
<td>SUCCESS</td>
<td>v1.2.0</td>
<td>Running</td>
</tr>
<tr>
<td>Fluentd Logging</td>
<td>SUCCESS</td>
<td>b637c60f1</td>
<td>Running</td>
</tr>
<tr>
<td>Linkerd Service Mesh</td>
<td>SUCCESS</td>
<td>v1.4.6</td>
<td>Running</td>
</tr>
<tr>
<td>ONAP Network Automation</td>
<td>SUCCESS</td>
<td>v1.11.1</td>
<td>Running</td>
</tr>
</tbody>
</table>

Last updated: 1 minute ago

Created for CNCF by Cross-Cloud CI
<table>
<thead>
<tr>
<th>Project</th>
<th>Build</th>
<th>Release</th>
<th>Deployments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td>v1.11.2</td>
<td>03a6d0b</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Prometheus</td>
<td>v2.3.2</td>
<td>c663477</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>CoreDNS</td>
<td>v1.2.0</td>
<td>b87ef6d1</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Fluentd</td>
<td>v1.2.4</td>
<td>fd8657fa</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Linkerd</td>
<td>v1.6.6</td>
<td>f96270c</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>ONAP</td>
<td>v1.11</td>
<td>8ad7083</td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

CI DASHBOARD: Overview

Last updated: 1 minute ago

Created for CNCF by Cross-Cloud CI
Installing terminal-table 1.8.8
Fetching jenkins-xcd_client 1.6.1
Installing jenkins-xcd_client 1.6.1
Installing vcr 4.6.0
Installing netmock 3.3.0
Installing netmock 3.3.0
Ran (5) complete 34 Gemfile dependencies, 0 gems not installed.
Ran (5) complete 34 Gemfile dependencies, 0 gems not installed.
Post-install message from aruba:
use $ ruby-1.8.7
* Make sure you add something like that to your 'Gemfile'. Otherwise you will
get cucumber > 2 and this will fail on ruby 1.8.7

gem 'cucumber', '>= 1.3.0'

With aruba >= 1.0 there will be breaking changes. Make sure to read
https://github.com/cucumber/aruba/blob/master/History.md for 1.0.0

$ cd...

$ /app/service/k8s/build_pipeline build status --integration=$INTEGRATION_NAME --release-type=$INTEGRATION_RELEASE_TYPE so

Build status: success

$ /app/service/k8s/build_pipeline download_container --integration=$INTEGRATION_NAME --release-type=$INTEGRATION_RELEASE_TYPE so

Package:

Pipeline #12567 from master

Commit 4005c6f8

Merge "Fix the structure in the doc for templates"
<table>
<thead>
<tr>
<th>Project</th>
<th>Build</th>
<th>Release</th>
<th>Deployments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kubernetes</td>
<td>v1.11.2</td>
<td>03a6d0b</td>
<td>SUCCESS</td>
</tr>
<tr>
<td></td>
<td>v1.11.2</td>
<td>03a6d0b</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Prometheus</td>
<td>v2.3.2</td>
<td>c663477</td>
<td>SUCCESS</td>
</tr>
<tr>
<td></td>
<td>v2.3.2</td>
<td>c663477</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>CoreDNS</td>
<td>v1.2.0</td>
<td>b83e801</td>
<td>SUCCESS</td>
</tr>
<tr>
<td></td>
<td>v1.2.0</td>
<td>b83e801</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Fluentd</td>
<td>v1.2.4</td>
<td>fd8871a</td>
<td>SUCCESS</td>
</tr>
<tr>
<td></td>
<td>v1.2.4</td>
<td>fd8871a</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Linkerd</td>
<td>1.4.6</td>
<td>b96277b</td>
<td>SUCCESS</td>
</tr>
<tr>
<td></td>
<td>1.4.6</td>
<td>b96277b</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>ONAP</td>
<td>v1.11</td>
<td>8ad7083</td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

CI DASHBOARD: Overview

Last updated: 1 minute ago

Created for CNCF by Cross-Cloud CI

Report bug
Waiting for 1 ready nodes. No ready nodes found, retrying.
No resources found.
Waiting for 1 ready nodes. No ready nodes found, retrying.
No resources found.
Waiting for 1 ready nodes. No ready nodes found, retrying.
No resources found.
Waiting for 1 ready nodes. No ready nodes found, retrying.
No resources found.
Waiting for 1 ready nodes. No ready nodes found, retrying.
No resources found.
Detection of 0 ready nodes, found 0 nodes out of expected 1. Your cluster may not be fully functional.
No resources found.

$ helm init --service-account tiller
Creating /root/helm
Creating /root/helm/repository/cache
Creating /root/helm/repository/local
Creating /root/helm/staging
Creating /root/helm/cache
Creating /root/helm/repository/repositories.yaml
Adding stable repo with URL: https://charts-stable.release.storage.googleapis.com
Adding local repo with URL: https://127.0.0.1:8879/charts
helm repo add stable https://stable releasestorage.googleapis.com
No Helm repo configured at /root/helm.

Tiller (the Helm server-side component) has been installed into your Kubernetes cluster.

Please note: by default, Tiller is deployed with an insecure "allow unauthenticated users" policy.
For more information on securing your installation use: https://docs.helm.sh/using-helm/Securing-your-Helm-installation

$ helm version
$ helm lint
$ helm rollback status = deployment/Tiller-deploy --namespace=kube-system
Waiting for deployment spec update to be observed...

COMMIT: 1bfa1a58: execution took longer than 30000 seconds.
### CI Dashboard: Overview

<table>
<thead>
<tr>
<th>Project</th>
<th>Build</th>
<th>Release</th>
<th>Deployments</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Status</td>
<td>Stable Head</td>
<td>AWS</td>
</tr>
<tr>
<td>Kubernetes Orchestra</td>
<td>SUCCESS</td>
<td>v1.11.2</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Prometheus Monitoring</td>
<td>SUCCESS</td>
<td>v2.3.2</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>CoreDNS Service Discovery</td>
<td>SUCCESS</td>
<td>v1.2.0</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Fluentd Logging</td>
<td>SUCCESS</td>
<td>v1.2.4</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>Linkerd Service Mesh</td>
<td>SUCCESS</td>
<td>1.4.6</td>
<td>SUCCESS</td>
</tr>
<tr>
<td>ONAP Network Automation</td>
<td>SUCCESS</td>
<td>v1.11</td>
<td>SUCCESS</td>
</tr>
</tbody>
</table>

Last updated: 1 minute ago

Created by Cross-Cloud CI
Job P67042 triggered about 5 minutes ago

Starting Run on display: -- with res 1080x1920mA
Executing robot tests at log level FINEST

OpenCOMP ETO
OpenCOMP ETO:Robot
OpenCOMP ETO:Robot:Testsuites
OpenCOMP ETO:Robot:Testsuites:Health Check : Testing ecmp components are...
Basic: P2Q: Health Check | PASS |
OpenCOMP ETO:Robot:Testsuites:Health Check : Testing ecmp components ...
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

OpenCOMP ETO:Robot:Testsuites | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

OpenCOMP ETO:Robot | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

OpenCOMP ETO | PASS |
1 critical test, 1 passed, 0 failed
1 test total, 1 passed, 0 failed

Output: /share/logs/EVOE5178/output.xml
Log: /share/logs/EVOE5178/log.html
I call this run successful...
Technology Overview
CI System Technology Overview

Unified CI/CD platform:
  • GitLab

Cross-cloud provisioning:
  • Terraform, Cloud-init, and per cloud K8s configuration

App deployments:
  • K8s manifest management with Helm

E2e tests:
  • Custom containers + Helm

Automated builds and deployments:
  • Git + per project yaml configuration
Dashboard Technology Overview

Frontend:
  • Vue.js

Status repository:
  • Elixir and Erlang

Automated builds and deployments:
  • Git + per project yaml configuration
How to Collaborate

Attend CI WG meetings:
• [https://github.com/cncf/wg-ci](https://github.com/cncf/wg-ci)
• 4th Tuesday of the month at 11:00am Pacific Time
• Next Meeting is on Tuesday, Sept 28th

Subscribe to the CNCF CI public mailing list:
• [https://lists.cncf.io/g/cncf-ci-public](https://lists.cncf.io/g/cncf-ci-public)

Create issues on GitHub:
• [https://github.com/crosscloudci/cross-cloud/issues](https://github.com/crosscloudci/cross-cloud/issues)

Join the #cncf-ci channel on slack:
• Request invite at [https://slack.cncf.io/](https://slack.cncf.io/)
• Cloud-native.slack.com
Connect with Cross-Cloud CI

@crosscloudci

@crosscloudci

crosscloudci@vulk.coop
For more details and an in-depth demo, please contact Dan Kohn & Cross-Cloud CI team at CNCF booth at #OSSNA18

Also presenting at:

- KubeCon + CloudNativeCon China
  - November 13-14, Shanghai