Evolving Cloud Native Landscape

Chris Aniszczyk, CTO (@cra)
https://cncf.io
https://linuxfoundation.org
Agenda

• Introduction
• What is CNCF?
• From Virtualization to Containers and Cloud Native
• Evolving Cloud Native Landscape
• VNFs to CNFs
• Q&A
Hi, I’m Chris Aniszczyk (@cra)

› CTO/COO, Cloud Native Computing Foundation (CNCF)
› Executive Director, Open Container Initiative (OCI)
› VP, Developer Relations, Linux Foundation (LF)

› In a previous life…
  › Director of Open Source (@Twitter) / Sr. Eng Manager
  › Co-Founder of the TODO Group
  › Co-Founder of EclipseSource (via Code9)
  › Open Source Committer (Gentoo, Fedora, etc)
  › Principal Software Engineer, Red Hat
  › Senior Software Engineer, IBM
Been a crazy ~3 years for me... Google Trends + CNCF

Kubernetes  OpenStack  Mesos  Docker

Swarm  Cloud Foundry

YOU FOUGHT IN THE CONTAINER WARS?
YES. I WAS ONCE A DEVELOPER, THE SAME AS YOUR FATHER

Drew Petersen
@KirbySaysHi
Regarding the Docker/Rocket stuff
5:26 PM - 2 Dec 2014

Follow
Cloud Native Computing Foundation (CNCF)

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

- Platinum members:
300+ Members and Growing

Platinum Members

Alibaba Cloud  AWS  Azure  Cisco  Dell Technologies  Docker  Fujitsu  Google Cloud  Huawei

IBM Cloud  Intel  JD.COM  Mesosphere  Oracle  Pivotal  Red Hat  Samsung

Gold Members

AT&T  Baidu  DigitalOcean  JFrog  NEC  NetApp  Salesforce  SumoLogic  SUSE  Tencent Cloud  ZTE

End User Members

Atlassian  Bloomberg  CapitalOne  Comcast  eBay  Denso  DiDi  Github  Goldman Sachs  Indeed  Intuit  JD.COM

Morgan Stanley  NAIC  NCSoft  Pinterest  Salesforce  Spotify  Showmax  Stitch  Two Sigma

End User Supporters

Adidas  Box  Cruise  Form3  Kuelap  Layer  Nasdaq  Shopify  Salesforce  Spredfast

TENX  ThredUp  Ticketmaster  Werkspot  Woorank

Academic/Nonprofit

CableLabs  Cloud Foundry  Golden Gate University  HCL  MIT

SquareSpace  Twilio  Twitter  WIKImedia Foundation  Yahoo! Japan

TWO SIGMA  Zalo  Zendesk
300+ Members and Growing (Silver 1)
300+ Members and Growing (Silver 2)
67 Companies in the End User Community

Plus 7 non-public members
Certified Kubernetes Conformance

- CNCF launched a software conformance program for Kubernetes
  - Implementations run conformance tests and upload results
  - New mark and more flexible use of Kubernetes trademark for conformant implementations
  - Taking submissions now for K8s 1.10 & 1.11
  - [https://www.cncf.io/certification/software-conformance/](https://www.cncf.io/certification/software-conformance/)
55 Certified Kubernetes Partners
Cloud native computing uses an open source software stack to:

- segment applications into microservices,
- package each part into its own container
- and dynamically orchestrate those containers to optimize resource utilization
Why You Should Host Your Project at CNCF

- A neutral home increases contributions
- Endorsement by CNCF’s Technical Oversight Committee
- Engagement with End User and Service Provider Communities
- Full-time press and analyst relations teams
- Tens of thousands of dollars per year in documentation, case study and other support services
- Maintain your committers and define your own governance, as long as its neutral
- Full-time staff eager to assist
- World-class events team, track at KubeCon + CloudNativeCon around the world, and custom events for your project
- Worldwide meetup groups
- Cloud resources for CI and scale testing
This landscape is intended as a map through the previously uncharted forest of cloud native technologies. There are many routes to deploying a cloud-native application, with CNCF projects representing a particularly well-trodden path. 

See the interactive landscape at cnfd.io
Try it now at https://l.cncf.io
Cloud Native Trail Map

Trail Map: lcnf.io

CLOUD NATIVE TRAIL MAP

The Cloud Native Computing Foundation CoMl has a large number of options. The Cloud Native Trail Map is a recommendation process for leveraging open-source, cloud-native technologies. As each step you choose a vendor-supported offering or do it yourself, and everything else in between, it’s optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer.

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider.

C. Join CNCF’s End User Community

For companies that don’t offer cloud native services externally, consider joining.

WHAT IS CLOUD NATIVE?

Cloud native technologies, such as containers and microservices, empower organizations to develop and deploy scalable, agile applications and services with ease. Many of these applications are microservices architectures. By taking into account these characteristics, both systems are designed to be resilient, elastic, and loosely coupled, via message-oriented abstractions and declarative APIs, thereby enabling effective, reliable automation. This allows engineers to observe the applications and to safely make incremental changes, and results in processes and workflows that fully take advantage of these environments and enormous cost.

The Cloud Native Computing Foundation seeks to drive adoption of these technologies by fostering an ecosystem of open-source, vendor-neutral projects that align with those objectives, and which are portable to public, private, and hybrid clouds. We demonstrate the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

1. CONTAINERIZATION

- Containerize data with Docker containers
- Any size application and dependencies (even FEP-11 code) running on an emulated container can be containerized
- Over time, you should expect benefits of a platform applications and writing future functionality as microservices

2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually perhaps, to production
- Setup automated rollouts, roll backs and testing

3. ORCHESTRATION

- Kubernetes is the main-heading orchestration solution
- You should select a Certified Kubernetes Distribution, Trusted Platform, or installer
- Use tools

5. SERVICE MESH AND DISCOVERY

- Venturi is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

6. NETWORKING

- To enable more reliable networking, use a CNF-compliant network project like CNI, Flannel, or Ivanovo

7. DISTRIBUTED DATABASE

- When you need more resiliency and scalability than you can get from a single database, where is good solution for moving MySQL at scale through sharding

8. MESSAGING

- When you need higher performance than Kafka, consider using gRPC. NATS is publish/subscribe message-oriented middleware

9. CONTAINER RUNTIME

- Hyperscale containerized container runtimes
- The most common, all of which are OCI-compliant, are containers, AK and CNCF

10. SOFTWARE DISTRIBUTION

- If you need to move software, consider creating an evaluative Notary, an implementation of The Update Framework.
Step 1: Containerization

• You can’t do cloud native without containerizing your application!
• Any size application will do!
• Containerize your legacy: Ticketmaster has a PDP-11 emulator running inside a container!

1. CONTAINERIZATION
• Normally done with Docker containers
• Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
• Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices
Step 2: CI/CD

• Setup Continuous Integration and Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production

• Setup automated rollouts, roll backs and testing

• CNCF doesn’t offer a CI/CD solution but many options out there: Jenkins, GitLab CI, Spinnaker, etc
Step 3: Orchestration

• Pick an orchestration solution
• Kubernetes is the market leader and you should select a Certified Kubernetes Platform or Distribution: https://www.cncf.io/ck
• However, other orchestration solutions exist: Mesos, Pouch, Nomad, Docker Swarm
Step 4: Observability

- Observability is table stakes for cloud native at scale: Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus (prometheus.io) for monitoring
- Fluentd (fluentd.io) for logging
- For tracing, look for an OpenTracing-compatible (opentracing.io) implementation like Jaeger (jaegertracing.io)
Step 5: Service Mesh

- Connects services together and monitors ingress from the Internet
- Service discovery, health checking, routing, load balancing
- Consider Envoy (envoyproxy.io), Linkerd (linkerd.io) and CoreDNS (coredns.io)
Step 6: Networking

- To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net
  - https://github.com/containernetworking/cni
### Step 7: Distributed Database

- When you need more resiliency and scalability than you can get from a single database, Vitess ([vitess.io](https://vitess.io)) is a good option for running MySQL at scale through sharding.

- TiKV is a distributed key value store: [https://github.com/tikv/tikv](https://github.com/tikv/tikv)
Step 8: Messaging

• When you need higher performance than JSON-REST, consider using gRPC: grpc.io

• NATS is a publish/subscribe message-oriented middleware: nats.io
Step 9: Container Runtimes

- You can use alternative container runtimes. The most common, all of which are OCI-compliant, are containerd (containerd.io), rkt (rkt.io) and CRI-O (cri-o.io).
- Alibaba’s PouchContainer runtime also makes for an option:
Step 10: Software Distribution

- If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.
- TUF is designed to work as part of a larger software distribution framework and provides resilience to key or server compromises. Using a variety of cryptographic keys for content signing and verification, TUF allows security to remain as strong as is practical against a variety of different classes of attacks. TUF is used in production by Docker, LEAP, App Container, Flynn, OTAInfo, ATS Solutions, and VMware.
- Harbor is a container registry
Cloud Native Trail Map Summary

- There are many paths to cloud native and the CNCF is doing our best to guide companies on their journey.

- The landscape and trail map are open source, please contribute issues and improvements: https://github.com/cncf/landscape
VNFs to CNFs
Network Architecture 1.0 (physical boxes, routers, etc)
Network Architecture 2.0:
Physical boxes converted to virtual machines called Virtual Network Functions (VNFs) running on VMs
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

- 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
Benefits of CNFs?

• Cost savings (with public, private, and hybrid clouds)
• Development velocity (cloud native)
• Resilience (to failures of individual CNFs, machines, and even DCs)
Challenges of evolving CNFs to VNFs

• 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
CNFs Discussion Areas

• 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](https://www.google.com) and [Jess](https://www.jess.com) that come into play when hosting untrusted, user-provided code
  – Possible use of isolation layers such as [gVisor](https://www.gvisor.dev) or [Kata](https://www.katacontainers.io)

• Scheduling container workloads with network-related hardware constraints (similar to what’s been done for GPUs)
  – Network-specific functionality like QOS
CNFs Demos

- **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
Getting Involved / Q&A
Contributing to CNCF for Networking and Telcos

• See the landscape
  – CNI: https://github.com/containernetworking/cni
  – Envoy: https://github.com/envoyproxy/envoy
  – Linkerd: https://github.com/linkerd/linkerd

• Kubernetes IoT/Edge WG
  – https://github.com/kubernetes/community/tree/master/wg-iot-edge

• Cloud-native Network Functions (CNFs)
  – #cnfs on slack.cncf.io
  – https://github.com/cncf/cnfs
CNCF MOOC and Online Training

• Free Introduction to Kubernetes self-paced course offered with edX

• Kubernetes Fundamentals course
  – Content maps to Certified Kubernetes Administrator (CKA) exam
  – $299, intermediate level

• Open source curriculum available for companies offering training
  – CKA Exam coupons available with a bulk discount
Online, Proctored Kubernetes Exams

- **Certified Kubernetes Administrator (CKA)**
  - Over 1,500 registrations already
  - [https://www.cncf.io/certification/expert/cka/](https://www.cncf.io/certification/expert/cka/)

- **Certified Kubernetes Application Developer (CKAD)**
  - Certifies that users can design, build, configure, and expose cloud native applications for Kubernetes
  - [https://www.cncf.io/certification/expert/cka/ckad/](https://www.cncf.io/certification/expert/cka/ckad/)

- **Both tests**
  - Tests consist of a set of scenarios to resolve from the command line over 3 hours; there is no multiple choice
  - Each exam is $300
  - Quarterly exam updates to match K8s releases
KubeCon + CloudNativeCon

• China
  – Shanghai: November 13-15, 2018
  – Sponsorships open

• North America
  – Seattle: December 10-13, 2018
  – Sponsorships open

• 2019
  – Barcelona: May 20-23, 2019
  – Shanghai: June 26-28, 2019
  – San Diego: November 18-21, 2019
Thank you! Q&A?

Join now: https://cncf.io/join

or

General Inquiries: info@cncf.io
PR: pr@cncf.io
Event Sponsorships: sponsor@cncf.io
Membership: memberships@cncf.io
A Brief History of the Cloud
Non-Virtualized Servers: Sun (2000)

- Launching a new application? Buy a new server; or a rack of them!
- Building block of your application is physical servers
Virtualization: VMWare (2001)

- Releases for server market in 2001
- Popularizes virtual machines (VMs)
- Run many VMs on one physical machine, meaning you can buy fewer servers!
- Architectural building block becomes a VM
IaaS: AWS (2006)

- Amazon Web Services (AWS) creates the Infrastructure-as-a-Service market by launching Elastic Compute Cloud (EC2) in 2006
- Rent servers by the hour
- Convert CapEx to OpEx
- Architectural building block is also a VM, called an Amazon Machine Image (AMI)
PaaS: Heroku (2009)

• Heroku popularizes Platform-as-a-Service (PaaS) with their launch in 2009
• Building block is a buildpack, which enables containerized 12-factor applications
  - The process for building the container is opaque, but:
  - Deploying new version of an app is just: git push heroku
Open Source IaaS: OpenStack (2010)

- OpenStack brings together an extraordinarily diverse group of vendors to create an open source Infrastructure-as-a-Service (IaaS)
- Competes with AWS and VMWare
- Building block remains a VM
Open Source PaaS: Cloud Foundry (2011)

- Pivotal builds an open source alternative to Heroku’s PaaS and launches the Cloud Foundry Foundation in late 2014
- Building block is Garden containers, which can hold Heroku buildpacks, Docker containers and even non-Linux OSes
Containers: Docker (2013)

- Docker combines LXC, Union File System and cgroups to create a containerization standard adopted by millions of developers around the world
- Fastest uptake of a developer technology ever
- Enables isolation, reuse and immutability
Cloud Native: CNCF (2015)

Cloud native computing uses an open source software stack to:
- segment applications into microservices,
- package each part into its own container
- and dynamically **orchestrate** those containers to optimize resource utilization
What Have We Learned?

• Core Building Block:
  - Servers ➔ Virtual Machines ➔ Buildpacks ➔ Containers

• Isolation Units
  - From heavier to lighter weight, in spin-up time and size

• Immutability
  - From pets to cattle

• Provider
  - From closed source, single vendor to open source, cross-vendor
What About PaaS?

• OpenShift, Huawei CCE, Deis, and Apprenda are examples of PaaS’s built on top of cloud native platforms

• Many new applications start out as 12-factor apps deployable on a PaaS
  – In time they sometimes outgrow PaaS
  – And some apps never fit a PaaS model

• PaaS on top of cloud native supports both