CNCF Cloud Native Definition

Status: Final

Owners: Brian Grant, Justin Garrison

Approved by TOC: [date]

Last update: May 19, 2018

Pull request: https://github.com/cncf/toc/pull/117

Final Draft:

Cloud native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.

The Cloud Native Computing Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor-neutral projects. We democratize state-of-the-art patterns to make these innovations accessible for everyone.

Justin's draft:

Cloud-native technologies empower organizations to build and run scalable applications in dynamic environments. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs are commonly used.

These techniques create resilient systems that are manageable, observable, and loosely coupled. Robust automation minimizes toil and enables engineers to make high-impact changes frequently and predictably.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by sustaining an ecosystem of open source, vendor-neutral projects. We democratize the state-of-the-art patterns to make these innovations accessible for everyone.

Thockin draft, picking nits:

Cloud-native technologies empower organizations to build and run scalable applications and services in dynamic modern environments such as public and private clouds. Examples include containers, service meshes, microservices, immutable infrastructure, and declarative APIs and configuration.

These tools enable developers to create resilient, elastic systems that are manageable, observable, and loosely coupled. Operators can build and employ robust automation that minimizes toil and enables them to make high-impact changes frequently and predictably.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, vendor-neutral projects that align with these objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

Eighth draft:

Cloud-native technologies empower organizations to develop, deploy, and operate scalable applications and services in dynamic, distributed environments, such as public and private clouds. Examples include containers, service meshes, microservices, immutable infrastructure, and declarative APIs and configuration.

These technologies enable engineers:

- to create systems that are resilient, elastic, manageable, observable, and loosely coupled,
- to employ automation that minimizes toil, and
- to make high-impact changes predictably and frequently.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, portable, vendor-neutral projects that align with these objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

Seventh draft:

Cloud-native technologies empower organizations to develop, deploy, and operate scalable applications and services rapidly and on modern cloud infrastructure. This includes any public cloud as well as other dynamic, distributed environments.

Engineers combine these technologies with design patterns to create systems that are resilient, elastic, manageable, observable, and loosely coupled. This enables automation that minimizes toil, and enables teams of engineers to make high-impact changes predictably and frequently.

Examples of cloud-native technologies and patterns include containers, service meshes, microservices, immutable infrastructure, and declarative APIs and configuration.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, portable, vendor-neutral projects that align with these objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

Draft 6.5

Cloud-native technologies empower organizations to deploy scalable, agile applications in dynamic cloud environments. Examples include containers, service meshes, microservices, immutable infrastructure, and declarative APIs and configuration.

These systems are designed to be resilient, manageable, observable, and loosely coupled. This enables automation that minimizes toil, and enables engineers to make high-impact changes predictably and frequently.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, portable, vendor-neutral projects that align with these

objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

Sixth+ draft:

Cloud-native technologies empower organizations to develop and deploy scalable, agile applications and services in dynamic, distributed environments, such as public and private clouds.

Cloud-native systems are designed to be resilient, elastic, manageable, observable, and loosely coupled. This enables automation that minimizes toil, and enables engineers to make high-impact changes predictably and frequently.

Examples of cloud-native technologies and patterns include containers, service meshes, microservices, immutable infrastructure, and declarative APIs and configuration.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, portable, vendor-neutral projects that align with these objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

NOTE: removed intent-oriented for reasons discussed in the comment thread

---

Fifth draft:

Cloud-native technologies, such as containers and microservices, empower organizations to develop and deploy scalable, agile applications and services in dynamic, distributed environments. Such systems are designed to be resilient, elastic, operable, observable, and loosely coupled via APIs. This enables reliable automation that minimizes toil, and results in processes that enable graceful online evolution.

The Cloud Native Computing Foundation seeks to drive adoption of these techniques by fostering an ecosystem of open-source, portable, vendor-neutral projects that align with these objectives. We democratize the state-of-the-art patterns and practices to ensure innovations remain open and accessible for everyone.

Fourth++ draft: (feedback will be addressed in a new draft, above)

Cloud-native technologies, such as containers and microservices, empower organizations to develop and deploy scalable, agile applications and services in highly dynamic, distributed environments. Such systems are designed to be resilient, elastic, and loosely coupled via manageable abstractions and declarative APIs. This enables effective, reliable automation that minimizes toil, and results in processes and workflows that allow operators to make impactful changes safely and take full advantage of these environments.

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

Third draft:

Cloud-Native technologies are designed to operate with high velocity at scale in dynamic and distributed environments, such as public clouds and software-defined data centers. Such Cloud-Native applications, services, platforms, and infrastructure are engineered to provide and/or enable self service and high levels of automation through techniques such as abstraction, operability, observability, resilience, agility, elasticity, and loose coupling. They utilize approaches such as declarative APIs and microservices, and include mechanisms such as application containers and service meshes.

The mission of the Cloud Native Computing Foundation is to advance the state of the art and drive adoption of Cloud-Native technologies by fostering an ecosystem of open-source projects that are portable, vendor-neutral, and interoperable through well defined interfaces.

Second draft:

The mission of the Cloud Native Computing Foundation is to drive the adoption of technologies designed for modern, dynamic, and distributed environments, such as public clouds and software-defined data centers. Cloud-native applications, services, platforms, and infrastructure are engineered to provide and/or enable operability, observability, elasticity, resilience, and

agility. The Foundation seeks to foster interoperable cloud-native projects and to advance the state of the art by sustaining open source projects that embody and/or support these attributes:

- **Operability**: Expose control of application/system lifecycle.
- **Observability**: Provide meaningful signals for observing state, health, and performance.
- **Resilience**: Fast automatic recovery from failures.
- **Agility**: Fast deployment, iteration, and reconfiguration.
- **Elasticity**: Grow and shrink available resources to meet fluctuating demand

Example technologies and patterns that can be used to implement the above attributes, such as declarative configuration, APIs, application containers, and service meshes, are discussed in more detail in Schedule A, below.

First draft:

The Foundation's mission is to create and drive the adoption of a new computing paradigm, dubbed Cloud-Native Computing, designed to facilitate a high velocity of change to applications, services, and infrastructure at scale in modern distributed environments such as public clouds and private data centers, while providing high degrees of security, reliability, and availability. To that end, the Foundation seeks to shape the evolution of the technology to advance the state of the art for application management and to foster an ecosystem of Cloud-Native technologies that are interoperable through well defined interfaces, and which are portable, vendor-neutral, and ubiquitous.

The following are some attributes of Cloud Native:

- Cloud-native services should enable self-service. For instance, cloud-native resources should be self-provisioned from an elastic pool that for typical, on-demand usage appears to be of unlimited capacity.
- Cloud-native environments are dynamic. They necessitate applications and services running in such environments to be automatically discovered, self-healing, and adaptable to frequent changes.
- Cloud-native applications, services, and infrastructure facilitate high-velocity management at scale via automation, which is enabled by exposing control of application and system lifecycle, supporting dynamic configuration, and providing

meaningful signals for observing state. In particular, resource usage is measured to enable optimal and efficient use.
- Cloud-native services and infrastructure are decoupled from applications, with seamless and transparent consumption experiences.

Defer to Schedule A:

Examples of mechanisms and patterns that promote or assist Cloud-Native approaches (without intending to imply they are exhaustive, exclusive, or required) include:

- Immutable infrastructure: Replace individual components and resources rather than updating them in place, which rejuvenates the components/resources, mitigates configuration drift, and facilitates repeatability with predictability, which is essential for high-velocity operations at scale.
- Application containers: Running applications and processes in containers as units of application deployment isolates them from their operational environments as well as from each other, facilitates higher levels of resource isolation, fosters component reuse, enables portability, increases observability, and standardizes lifecycle management.
- Microservices: Loosely coupled microservices significantly increase the overall agility and maintainability of applications. Microservices provide a model for scaling from small to large organizations without slowing down software delivery.
- Service meshes: Service meshes decouple service access from the provider topology, which reduces the risk of operational changes, and support inter-component observability.
- Declarative configuration: Intent-oriented configuration lets users focus on the What rather than the How, and reserves latitude for automated systems achieve the desired state.
- ~~Event-driven execution: Enables agile, reactive automated processes, and facilitates systems integration.~~

As new Cloud-Native techniques and technologies emerge, they will be incorporated into the Foundation’s portfolio of recommended practices, processes, and projects.

Original text:

1. Mission of the Cloud Native Computing Foundation.
The Foundation's mission is to create and drive the adoption of a new computing paradigm that is optimized for modern distributed systems environments capable of scaling to tens of thousands of self healing multi-tenant nodes.
Cloud native systems will have the following properties:

(a) Container packaged. Running applications and processes in software containers as an isolated unit of application deployment, and as a mechanism to achieve high levels of resource isolation. Improves overall developer experience, fosters code and component reuse and simplify operations for cloud native applications.
(b) Dynamically managed. Actively scheduled and actively managed by a central orchestrating process. Radically improve machine efficiency and resource utilization while reducing the cost associated with maintenance and operations.
(c) Micro-services oriented. Loosely coupled with dependencies explicitly described (e.g. through service endpoints). Significantly increase the overall agility and maintainability of applications. The foundation will shape the evolution of the technology to advance the state of the art for application management, and to make the technology ubiquitous and easily available through reliable interfaces.

…

Schedule A: Initial CNCF Scope Vision

The overarching intent behind the cloud native computing foundation is to support and accelerate the adoption of 'cloud native computing'. What follows below is an initial scope intended to articulate core concepts of 'cloud native computing' that the CNCF will strive to implement. This initial scope shall become a living document posted to the CNCF website.

CNCF's community believe there are three core attributes to cloud native computing:

- Container packaged and distributed.
- Dynamically scheduled.

- Micro-services oriented.

A cloud native computing system enables computing that builds on these core attributes, and embraces the ideals of:

- Openness and extensibility.
- Well-defined APIs at borders of standardized subsystems.
- Minimal barriers to application lifecycle management.

<diagram omitted>

When successful, the cloud native computing foundation will establish:

1. Standardized interfaces between subsystems.
2. A standard systems architecture describing the relationship between parts
3. At least one standard reference implementation of each sub-system.
4. Thinking about adding extensible architecture that end users can extend, replace or change behavior in every layer of the stack for their purposes.

The suggested set of areas/sub-systems:

- Application definition and orchestration
    - standard service definition
        - define a standard distributed system service that can be deployed
    - composite application definition
        - a standard model for packaging and shipping an app of many parts
    - orchestrator
        - a standard framework to deploy and manage an app of many parts
        - providing workload/job specific orchestration and control
- Resource scheduling

        - map containers to nodes (in either a cluster, or single node environment)
        - manage the life-cycle, scaling and health of containers and container groups
        - handle active scaling of containers
        - interface with infrastructure provisioning to request more resources as needed (either bare metal, private cloud, or public cloud)
        - interface with SDN and SDS to map to storage/network
- Distributed systems services
        - a standard set of services that are not bound to a single node
            - supporting application use cases
            - naming/discovery
            - locking/quorum
            - state management/sharding
            - logging/monitoring
        - a minimum atom of consumption for software (i.e. the model to find and consume something)
            - within the cluster
            - between clusters
            - from outside the cluster
        - A local agent that integrates CNCF into a local computing environment
            - maintains the lifecycle of containers on a node
            - monitors the health of container on a node
        - Compute node definition
            - a standard definition for what the minimum set of services on a compute node are (the actual node distribution is out of scope)
        - Infrastructure provisioning and integration
            - provide a standard interface and plugin model to request additional infrastructure
        - software defined datacenter integration
            - provide a standard interface and plugin model for network (SDN) and storage (SDS) integration with clusters
        - services (common capabilities made optionally available to all CNCF sub-systems)

        - Distributed state and scheduling management. Provide mechanisms to robustly handle cluster state.
        - Distributed control plane technology. Support robust command and control of systems across failure domain boundaries.
        - API server. Create a common mechanism to support REST based access to core services.
    - Tools and Visualization
        - A single ‘pane of glass’ to view and control distributed systems
        - Monitoring and operations capabilities

External systems. These systems are not considered part of the project, but should interface cleanly with the CNCF subsystems, and CNCF should work with the the appropriate communities to support standardization.

- Compute Node OS. The node (host OS for container applications) is considered out of scope of this effort. We should however work closely with OS providers to ensure that the minimal node specification (i.e. what local services a cloud native application requires) are well defined and standardized across the emerging lightweight container nodes.
- Container runtime and specification. This is critical, but should be driven through the OCI. This community must interface with the OCI group to ensure that requirements for cloud native operation are met, including where applicable, signing, verification and distribution.
- Identity management systems. CNCF should integrate with identity management systems, but subject to TOC guidance, not define a new model.

For further discussion
The following are considered out of scope for now, but might need to be included later.

- Application environment. Crafting a well formed, minimal userland environment for cloud native applications.

- OCI topics (container format and runtime). The goal is to integrate with the OCI group.

Extensibility Model

Any given sub-system should exist behind an open REST based API. Vendors should be able to provide alternate implementations of any given subsystem to optimize if for a given workload, or to add value through commercial technology. The models, and APIs, defined by the CNCF should be extensible such that vendors can add additional capabilities. However, interoperability across implementations is critical and therefore extensions should be optional.

Qualification Model

CNCF imagines a qualification model through API standardization, and rich integration testing. The community will produce a trademarkeable brand and reserve use of the mark for systems that meet integration requirements.

Three qualification levels will likely exist:

- Trademark Implementation. All relevant technologies are built from code housed in CNCF repositories.
- Trademark Compatible. Passing all integration test to and demonstrating deep semantic compatibility with ‘Trademark Implementation systems.
- Trademark version X API compliant. Compliant with the version X API.