Cloud-Native Applications 101

Table of Contents

Cloud-Native Applications 101

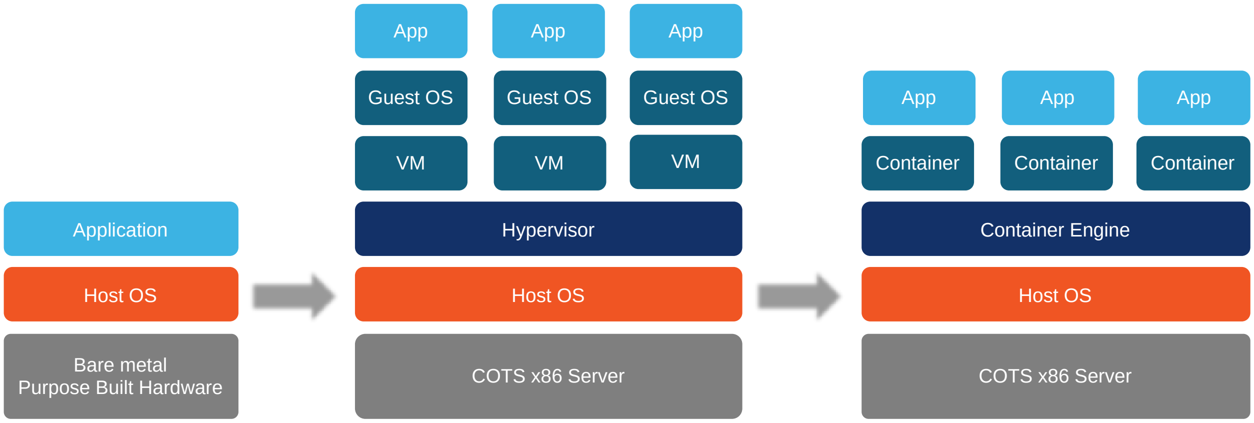
Introduction

Cloud-native is an approach in software development that utilizes cloud computing to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Cloud computing has become the norm, and it is now necessary that software-based applications are developed to run on cloud infrastructure that could be hosted on-premises or in an operator's owned data center, public cloud or hybrid cloud. In addition, service providers can take advantage of the capabilities offered by the cloud infrastructure to deliver service in an efficient, seamless and resilient way.

Cloud-native is not just about the technology change but also a mindset and culture change in the way software is developed, deployed and maintained. Whereas the waterfall model was a norm before, the cloud-native model now truly enables agile development.

Virtualization has evolved over time. Especially in telecom networks, network functions were running on purpose-built, hardware-based functions. But in order to reduce CAPEX and OPEX, operators then evolved the network functions to NFV with VM-based infrastructure. Virtualization helped in reducing overall TCO and increasing operator profitability, but still there were certain challenges with NFV. Now they have evolved to CNFs with cloud-native applications, which are gaining prominence given the associated benefits such as increased agility, improved innovation and shortened time to market for new features and services.

Cloud-native virtualization has been in use by several big technology companies like Google, Netflix and others. Some of these tech giants have been key contributors to the open source community in terms of seeding the code for some of the very popular cloud-native open source tools.

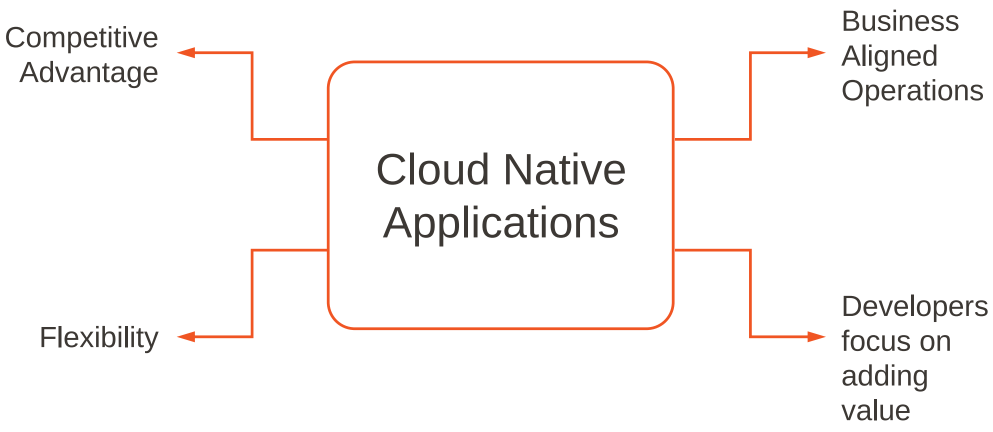


Benefits and Advantages

Traditional monolithic applications were designed with the purpose-built hardware environment in mind, which was later carried forward to NFV VM-based infrastructure. Monolithic applications are complex and stateful. Though modular internally, they have tight couplings or dependencies between those modules. Management, deployment, and scaling up/down is difficult and inefficient. In order to make them run on the cloud, there is a separate support or qualification needed for each cloud or stack.

In contrast, Cloud-native apps are stateless, easier to scale in/out and can be deployed in any cloud. They consist of loosely coupled microservices which are independently deployable, scaled, upgraded and downgraded. Cloud-native apps are efficient in resource requirements given the nature of environment they need. They provide the benefit of developing once deployed anywhere (bare metal servers, VMs, private cloud, public cloud or hybrid cloud).

Cloud-native offers benefits and advantages. The benefits are there from both the application user (such as the service provider) perspective as well as from the application developer (such as solution vendor) perspective.



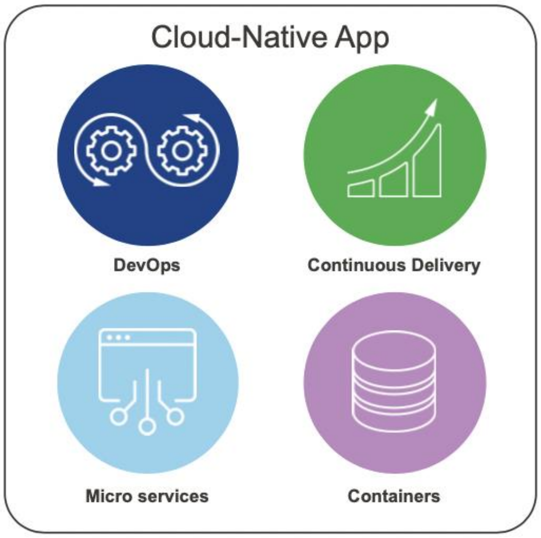
From an application user perspective, cloud-native offers a competitive advantage as it allows the user to be more agile and innovative. Cloud-native apps with Continuous Integration/ Continuous Delivery (CI/CD) capability allow the opportunity to try out new features and validate and iterate before launching them widely. This isn't easy with traditional monolithic apps, and they take lot of time. Cloud-native apps offer flexibility in terms of being able to be deployed on any type of cloud infrastructure, private or public. They provide elasticity in order to scale the application as per the load situation – thus, increasing resource (compute, storage and networking) utilization and power consumption efficiency. Faster recovery for applications is supported given that containers are lightweight, and instantiating and running containers is faster than running VMs.

With CI/CD and automation, all infrastructural and logistical aspects like upgrade/downgrade or new feature launch will be automated, helping to improve productivity and allow more focus on the core business goals.

From the application development perspective, cloud-native allows developers to focus on the application logic to deliver value to customers rather than worry about the infrastructure and environment given the capability of "build once, deploy anywhere" with cloud-native apps. It provides infrastructure abstraction which is a huge value-add. Further, it helps to reduce application complexity and simplifies dependencies between different teams, thus improving the collaboration.

What are Cloud Native Applications?

There are many interpretations of cloud-native. Some people just consider containers support as cloud-native; some others consider support for microservice as cloud-native.



Source: Pivotal.io

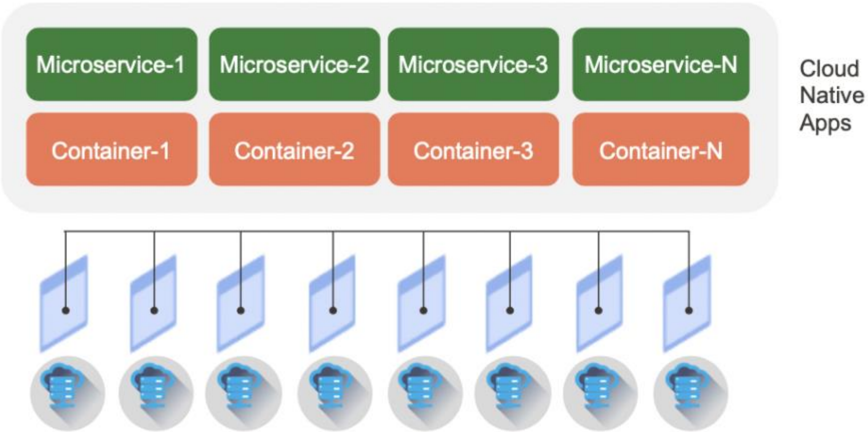
Strictly speaking, a cloud-native app needs to have support for these four key tenets:

1. Microservices
2. Containers
3. DevOps
4. CI/CD

Support for all of these key tenets is needed in order to truly deliver the benefits and advantages discussed earlier. Agile development talks about building software incrementally, shipping the functionality incrementally, getting feedback and iterating. Cloud-native development truly enables this real goal of agile development. Without this, there are challenges in executing agile in truest sense. These four tenets are intertwined, and all together deliver and execute the benefits expected out of a cloud-native application.

Microservices

Microservices is a technique that arranges an application as a collection of loosely coupled services. Basically, an application is decomposed into several microservice components. They communicate with each other over a network in order to fulfill a goal. A microservice is a self-contained piece of business functionality with clear interfaces exposed for others to use its services.



Microservices are autonomously developed, independently deployable, decentralized and built and released with automated processes.

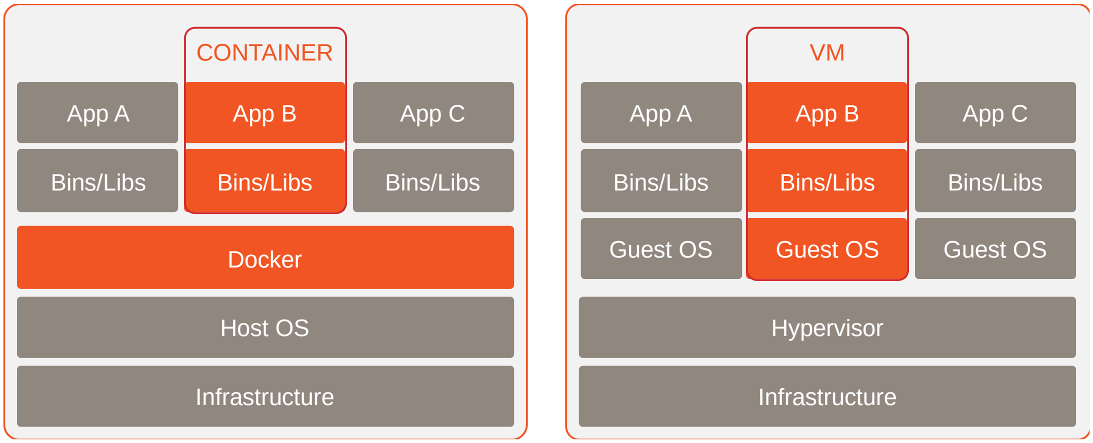
Microservice decomposition helps to reduce application complexity by breaking the application into smaller, self-contained components called microservices, each owned by a separate team. Each microservice has a well-defined API exposed for others to use the services offered by the microservice. This helps simplify dependencies between microservices, thereby improving the collaboration between teams.

There is no standard way of decomposing any application, but there are microservice characteristics defined by experts like Martin Fowler which act as some of the guidelines to perform the decomposition. Some of them are:

- Highly maintainable and testable
- Loosely coupled
- Independently deployable
- Organized around business capabilities
- Owned by a small team

Containers

A container image is a lightweight, standalone, executable package of a piece of software that includes everything needed to run it: code, runtime, system tools, system libraries, environment.



CONTAINERS VIRTUAL MACHINES

Containers offer both efficiency and speed compared with standard virtual machines. Using operating system level virtualization, a single OS instance is dynamically divided among one or more isolated containers, each with a unique writable file system and resource quota. Container images are typically in MBs and start almost instantly. Container Engine, like Docker, is a lightweight layer that enables running the containers.

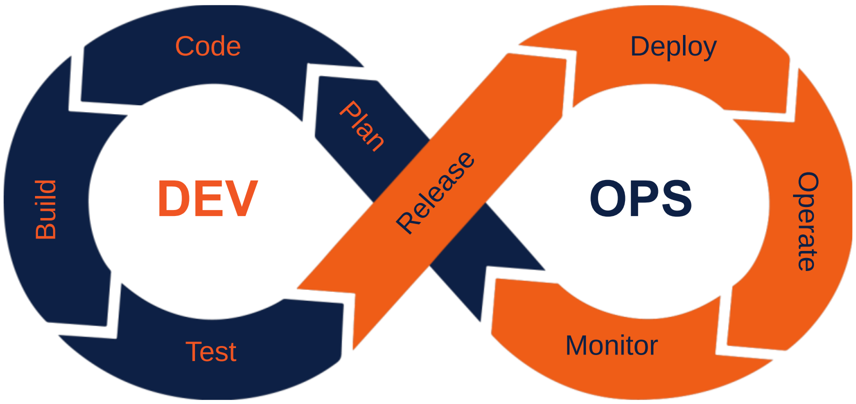
Whereas VMs are an abstraction of the physical hardware turning a physical server into many servers, hypervisors allow multiple VMs to be created on a physical server. Each VM in turn then includes a full copy of the OS, called Guest OS, one or more apps and necessary binaries and libraries – taking up to several GBs of memory. VMs are heavy and slow to start and may take seconds or minutes.

One of the key aspects with containers is container management. Container management is the process of managing the creation, deployment, scaling, availability, and destruction of software containers. Kubernetes (K8s) is one of the most popular container management tools. Kubernetes was originally designed by Google and is now maintained by the Cloud Native Computing Foundation. It aims to provide a "platform for automating deployment, scaling, and operations of application containers across clusters of hosts."

DevOps

DevOps strategy focuses on an organization's capability for continuous software delivery that enables customers to seize market opportunities and reduce time-to-customer feedback.

DevOps is a set of practices that combines software development (Dev) and Operations (Ops), which aims to shorten the systems development life cycle and provide continuous delivery with high software quality. It is intended to reduce the time between committing a change to a system and the change being placed into normal production, while ensuring high quality. Several DevOps aspects came from agile methodology.



Traditionally a solution vendor has two separate teams for development and operations, both working in silos, each having their own different environment for deployment or testing models. This leads to lot of back and forth between the two teams and slows down the process.

With the capability of "build once deploy anywhere" found with cloud-native, it becomes easier to remove the silos and simplify development, which expedites the operations process. DevOps is easily enabled in combination with containers, microservices and CI/CD with the capabilities provided by each of the components.

Doing DevOps with traditional monolithic applications is very complex due to applications being complex, big and heavy. Upgrade/downgrade is a very time-consuming and complex operation. Cloud-native overcomes these challenges using lightweight, faster containers, each hosting a component of the application and all done in an automated way in less time.

Continuous Integration/Continuous Delivery/Deployment

Continuous Integration (CI): The practice of merging all developers working code to a shared mainline several times a day. Every merge is validated before merging to catch any issues. This is in order to ensure that there are fewer integration issues and allows members in a team to collaboratively work together to deliver high quality software rapidly.

Continuous Delivery/Deployment (CD): The practice in which teams produce software in short cycles, ensuring that the software can be reliably released at any time, which can then be deployed as well. The intent is to build, test, and release software rapidly and frequently. This helps reduce the cost, time, and risk of delivering changes by allowing for more incremental updates to applications in production.

CI/CD is faster and easier with cloud-native applications as each microservice, a smaller component of the whole application, is released, managed and deployed independently. Doing CI/CD for a traditional monolithic application is slow and complex.

Real World Reference

Cloud-native based virtualization has been in use by several big technology companies like Google, Netflix and others. Some of these tech giants have been key contributors to the Open Source community in terms of the seeding the code for some of the very popular cloud native open source tools.

Netflix started working on cloud-native in 2009 after they faced a severe outage with their traditional monolithic solution. Adrian Cockcroft, who helped lead Netflix's migration to cloud-native, said, "*Cloud native architectures take full advantage of on-demand delivery, global deployment, elasticity, and higher-level services. They enable huge improvements in developer productivity, business agility, scalability, availability, utilization, and cost savings.*"

Verizon recently announced that it is working to bring a truly cloud-native stack to its 5G network. A complete end-to-end mobile network, including RAN and Core, will be built on the cloud-native stack.

Summary

Mobile network operators are facing strong pressures to evolve their businesses and operations for profitable growth and differentiation. Explosive demand for bandwidth persists, while ARPU pressures mount due to competition and limited service innovation. Addressing these challenges requires industry innovation focused on delivering new revenue-generating services based on a more flexible and agile service delivery environment, while reducing CAPEX and OPEX.

Virtualization and software-based solutions have become the norm and need of the hour. Cloud-native is the current and the future of the software-based applications and the way to develop, manage and deploy applications. Cloud-native applications come with several benefits in comparison to monolithic applications. Key benefits include competitive advantage, business-aligned operations, and flexibility, so that developers can focus on adding true value.