WHAT IS CLOUD-NATIVE?

Becoming “cloud native” is often cited as the end goal for migrating or building applications today. But depending on who you ask, you’ll probably get a lot of different definitions of what exactly it means. Overall, the majority boils down to this: Cloud native is an approach to building and running scalable applications to take full advantage of cloud-based services and delivery models.

WHAT IS A CLOUD-NATIVE APPLICATION?

A cloud-native application is specifically designed from the ground up to take advantage of the elasticity and distributed nature of the cloud. To better understand what a cloud-native application is, it’s best to start with what it’s not—a traditional, monolithic application.

Monolithic applications function as a single unit, often with custom-built operation systems, middleware, and language stacks for each application. Most scripts and processes are also purpose-built for the build, test, and deployment. Overall, this application architecture creates close dependencies, making it more difficult to change, test, deploy, and operate systems as they grow over time. What starts out as simple to design and deploy soon becomes complex, hard to evolve, and challenging to operate.

By comparison, cloud-native applications make the most of modern infrastructure's dynamic, distributed nature to achieve greater speed, agility, scalability, reliability, and cost efficiency.

Cloud-native applications are typically broken down into multiple, self-contained services through the use of technologies and methodologies, namely DevOps, continuous delivery and continuous integration, containers, microservices, and declarative APIs. This enables teams to deploy and scale components independently, so they can make updates, fix issues, and deliver new features without any service interruption.

CLOUD-NATIVE PILLARS

There are various ways to create a cloud-native architecture, but the goal is always to increase software delivery velocity and service reliability and to develop shared ownership among software stakeholders.
Still, the fundamentals of cloud-native architectures are based on five core pillars:

Microservices:

Almost all cloud architectures are based on microservices, but the key benefit they deliver is composability—breaking down an application into a collection of smaller, lightweight services that can easily be composed and connected to each other via application programming interfaces (APIs). For example, an ecommerce application might be composed of a specific service for the shopping cart, another for payment, and another one that communicates with the back end about inventory management. Composability also enables teams to swap and re-compose components to meet new business requirements without disrupting another part of the application.

Containers and orchestration:

Containers are lightweight, executable components that contain all the elements needed—including app source code and dependencies—to run the code in any environment. Containers deliver workload portability that supports “build once, run anywhere” code, making development and deployment significantly easier. They also help to reduce the chance of friction between languages, libraries, and frameworks since they can be deployed independently. This portability and flexibility makes containers ideal for building microservices architectures.

Container orchestration is also essential as the number of microservices grows to help manage containers so they can run smoothly as an application. A container orchestration platform like Kubernetes provides oversight and control of where and how containers run, repair any failures, and balance load between containers.

DevOps:

Cloud-native application development requires shifting to an agile delivery methodology like DevOps, where developers and IT operations teams collaborate to automate infrastructure and software delivery processes. DevOps allows development and operations teams to communicate more closely and come together around a shared purpose, creating a culture and environment where applications can be built, tested, and released faster.

Continuous integration and continuous delivery (CI/CD): Automation can repair, scale, and deploy systems much faster than people. CI/CD pipelines help automate the build, testing, and deployment of application changes without the need to schedule downtime or wait for a maintenance window. Continuous delivery ensures that software releases are more reliable and less risky, allowing teams to deliver new services and features more rapidly and frequently.

Cloud-native services:

Cloud-native services and technologies help you build, run, and deploy scalable applications in any environment. While your customers and business users benefit from a regular application, cloud-native services operate behind the scenes to keep things running smoothly.

For example, cloud-native services might describe the as-a-service offerings from cloud service providers (for example, Iaas, PaaS, and SaaS service models), the microservices of an application, and the APIs that connect and enable communication between services.

WHAT IS THE DIFFERENCE BETWEEN CLOUD AND CLOUD NATIVE?

There is actually a difference between cloud and cloud native. Cloud refers to cloud computing, where companies or individuals pay to access computing resources as an on-demand service.
While it is often used as a catch-all description for the tools and techniques used to develop software in the cloud, the term “cloud native” isn’t solely about cloud adoption. Instead, it refers to how applications are built and delivered, rather than just where they are deployed. In some cases, an application may not even run in the cloud. It’s possible to build applications with cloud-native principles and run it on-premises or in hybrid environments.

Cloud-native challenges

Despite the many cloud-native benefits, this model does come with some trade-offs that should be considered. Cloud-native computing is not always straightforward to implement as beyond adopting new tools and technologies, it also requires cultural shifts to make its use successful.

Some common cloud-native challenges include:

Dealing with distributed systems and many moving parts can be overwhelming if you don’t have tools or processes in place to manage development, testing, and deployment

Increased operational and technology costs without the right cost optimization and oversight in place to control the use of resources in cloud environments

Lack of existing technology skills to work with and integrate a more complex technology stack

Resistance to the cultural shifts needed to implement cloud-native technologies and DevOps best practices

Difficulty communicating cloud-native concepts to gain support and buy-in from non-technical executives

However, none of the above is unmanageable with the right expertise and strategy. For example, adopting a simple “lift and shift” approach to migrating to the cloud is a good place to start, but it won’t provide many of the cloud-native benefits listed above. Many organizations end up stalling out at this stage because they haven’t anticipated the expense and complexity of re-architecting to a cloud-native architecture.
We recommend not treating cloud native as a multi-year, big-bang project. Instead, it should be considered an ongoing journey of constant iteration to learn and improve as you go.

Taken from: https://cloud.google.com/learn/what-is-cloud-native