4.1 Defining cloud-native

Cloud-native concepts existed before the term itself came into use. The beginning of cloud-native was when public cloud vendors began providing easy and affordable access to elastic instances of compute power. How can you write applications to capitalize on the flexibility of this new infrastructure, and what business benefits can you achieve?

Cloud-native methods and technology are still evolving, but the core business objectives that cloud-native applications set out to achieve remained the same. The business benefits of cloud-native are:

- Agility: Enable rapid innovation that is guided by business metrics.
- Resilience: Target continuous availability that is self-healing and downtime-free.
- Scalability: Provide elastic scaling and limitless capacity.
- Cost: Optimize the costs for compute and human resources.
- Portability: Enable free movement between locations and providers.

Note that the “cloud” in cloud-native refers to the public cloud and the infrastructure that is rapidly self-provisioned, elastically scalable, and has apparently limitless capacity. Cloud-native solutions may be built on-premises or in dedicated environments, but it is the cloud-based approach to platform provisioning that is important.

Important: Microservices are associated with cloud-native, but they are not equivalent. Microservices architecture is a concrete example of how to build applications in a cloud-native style.

A key reference is the principle of 12-factor applications. Although the 12 factors make no claim to be all encompassing guidelines for cloud-native, they are one of the earliest examples of how to write applications with a cloud-native intent.

“The twelve-factor app is a methodology for building software-as-a-service (SaaS) apps that:

- Use *declarative* formats for setup automation, to minimize time and cost for new developers joining the project;
- Have a clean *contract* with the underlying operating system, offering *maximum portability* between execution environments;
- Are suitable for *deployment* on modern *cloud platforms*, obviating the need for servers and systems administration;
- *Minimize divergence* between development and production, enabling *Continuous Deployment* for maximum agility;
- And can *scale up* without significant changes to tooling, architecture, or development practices.”[1]

Despite its age (2011), this list encapsulates many of the key concepts and guidelines for cloud-native, and the 12 factors themselves still have relevance even if the specific terms we use have evolved in the past few years. It is a useful backdrop, and we describe the 12 factors in 4.3, “Twelve-factor apps” on page 100.

We should recognize the importance of contributions from early adopters in cloud-native.

[1] Source: https://www.12factor.net/