What is cloud-native? The modern way to develop software

Cloud-native computing takes advantage of many modern software development techniques including microservices, containers, CI/CD, agile methodologies, and devops.

By Scott Carey
UK Group Editor, InfoWorld
AUG 17, 2021 3:00 AM PDT

The term "cloud-native computing" has emerged as a catch-all for the various tools and techniques required by software developers to build, deploy, and maintain modern software applications on cloud infrastructure. Here, we define the term, survey the cloud-native landscape, and identify some of the advantages and pitfalls of going cloud-native.

[Also on InfoWorld: How to choose a cloud database]

Table of Contents

Cloud-native definition

Cloud-native is a modern approach to building and running software applications that exploits the flexibility, scalability, and resilience of cloud computing. Cloud-native encompasses the various tools and techniques used by software developers today to build applications for the public cloud, as opposed to traditional architectures suited to an on-premises data center.

The cloud-native approach to building and running software was pioneered by a group of companies commonly referred to as "born in the cloud" — such as streaming giants Netflix and Spotify, ride-hailing company Uber, and accommodation booking platform Airbnb. The cloud-native approach has since been adopted by other companies looking for similar digital agility and disruptive competitive advantage.

The Cloud Native Computing Foundation (CNCF) defines cloud-native a little more narrowly, focusing on application containerization — where applications are broken down into microservices and packaged in lightweight containers to be deployed and orchestrated across a variety of servers.

In the CNCF's own words: "Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds."

Cloud-native app development typically includes marrying microservices, cloud platforms, containers, Kubernetes, immutable infrastructure, declarative APIs, and continuous delivery technology with techniques like devops and agile methodology.

Cloud-native landscape

This shift in popular software development techniques has seen a new landscape of predominantly open source tools emerge. The CNCF maintains an interactive graphic of this ecosystem.

There are four layers to cloud-native computing that are important to understand:

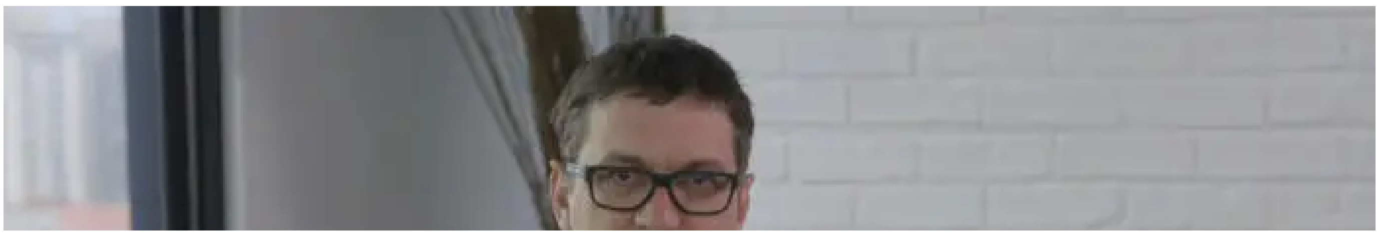
- **The application definition and development layer.** The top layer of the cloud-native stack focuses on the tools used by developers to build applications, such as databases, messaging systems, container images, and continuous integration and continuous delivery (CI/CD) pipelines.
- **The provisioning layer.** The provisioning layer of the cloud-native stack includes anything required to build and secure the environment where an application will run, ideally in a repeatable fashion. In the cloud-native world, this typically involves treating infrastructure as code, storing images in a repository, automating builds, and addressing application security needs with vulnerability scanning, key and policy management, and authentication tools.
- **The runtime layer.** The runtime layer concerns anything associated with the running of a cloud-native application, such as the container runtime—which still tends to be Docker—as well as storage and networking.
- **The orchestration and management layer.** The orchestration and management layer brings together the tools required to deploy, manage, and scale containerized applications, including orchestration and scheduling. In most cases, that means Kubernetes—as well as service discovery, service proxy, API gateway, and service mesh.

Outside these layers it is also important to implement observability practices, so all these services are monitored effectively. Some organizations also opt to bring their stack together into a self-service internal developer platform or to purchase an opinionated platform as a service (PaaS) from a vendor to ease adoption for developers.

[Also on InfoWorld: How Kubernetes works]

Related video: What is the cloud-native approach?

In this 60-second video, learn how the cloud-native approach is changing the way enterprises structure their technologies, from Craig McLuckie, founder and CEO of Heptio, and one of the inventors of open source Kubernetes.



The advantages of cloud-native vs. on-premises architectures

Cloud-native application development requires a very different architecture than traditional enterprise applications, which would typically run in an on-premises data center. Here are some key differences and the advantages cloud-native applications bring over traditional app dev models.

Languages. On-premises apps written to run on company servers tend to be written in traditional languages, like C/C++, C#, and enterprise Java. Cloud-native apps are more likely to be written in a web-centric language, like HTML, CSS, Java, JavaScript, .NET, Go, Node.js, PHP, Python, and Ruby. Working with modern languages and platforms can help when it comes to attracting the best engineers to work at your organization.

Updatability. Cloud-native apps are built to be highly available, resilient, and regularly updatable, whereas on-premises applications are typically updated once or twice a year using a waterfall methodology. Cloud-native computing's updatability provides a productivity boost for development teams to focus on their competitive advantage and deliver new features to customers more frequently than before.

Elasticity. Cloud-native applications typically take advantage of the elasticity of the cloud by flexing consumption depending on demand, whereas an on-premises application would require the physical provisioning of additional infrastructure to scale effectively. This also has cost implications, as the cloud allows you to pay for what you use and avoid costly overprovisioning of your own infrastructure—at least in theory.

Multitenancy. A cloud-native app has no problem working in a virtualized space and sharing resources with other applications using a multitenant model. This brings a clear efficiency boost to development teams.

Downtime. The cloud offers greater redundancy due to the scale and geographical spread of data centers managed by the hyperscale cloud vendors, so outages can be better managed by quickly redirecting traffic to another region and avoiding costly downtime.

Automation. Cloud-native techniques open up a wealth of automation opportunities for engineers to build once and move on to other more pressing challenges.

Stateless. Cloud-native applications tend to be stateless, in that they don't carry over saved data from one session to another. This model opens up the opportunity to scale easily across multiple servers, cache more easily for a performance boost, use less storage, and avoid that dreaded vendor lock-in by not being attached to a specific server.

Cloud-native challenges

Trying to lift and shift an existing on-premises application to be cloud-native without making architectural challenges is a common mistake, but rearchitecting something for the cloud is also a significant engineering challenge in itself.

Finding the right skills mix to do this, adapting to a cloud-centric security model, and managing the changing cost profile of a cloud environment all remain key challenges for organizations looking to go cloud-native.

[Keep up with the latest developments in software development. Subscribe to the InfoWorld First Look newsletter]

Still, developers should look to embrace cloud-native as an organizational principle, either by building new applications for the cloud or by breaking up existing monolithic applications into microservices so that they are better suited to a cloud environment.

This will require a significant mindset shift away from traditional waterfall deployments to more agile development principles like minimum viable product (MVP) development, embracing automation, multivariate testing, rapid iteration, observability, and working closely with the operations team in a devops model.

Learn more about related cloud-native technologies:

- Platform-as-a-service (PaaS) explained
- Multicloud explained
- Agile methodology explained
- Agile development best practices
- Devops explained
- Devops best practices
- Microservices explained
- Microservices tutorial
- Docker and Linux containers explained
- Kubernetes tutorial
- CI/CD (continuous integration and continuous delivery) explained
- CI/CD best practices

Scott Carey is the Group Editor for IDG UK enterprise titles, writing primarily for InfoWorld.

Follow