

What is Cloud Native? - Cloud Native Applications Explained - AWS

What is Cloud Native, how and why businesses use Cloud Native Applications, and how to use Cloud Native Applications with AWS.

7 min. read · [View original](#)

What is Cloud Native?

Cloud native is the software approach of building, deploying, and managing modern applications in cloud computing environments. Modern companies want to build highly scalable, flexible, and resilient applications that they can update quickly to meet customer demands. To do so, they use modern tools and techniques that inherently support application development on cloud infrastructure. These cloud-native technologies support fast and frequent changes to applications without impacting service delivery, providing adopters with an innovative, competitive advantage.

How does a cloud-native approach benefit businesses?

Organizations gain competitive advantages in various ways when they build cloud-native software applications.

Increase efficiency

Cloud-native development brings along agile practices like DevOps and continuous delivery (CD). Developers use automated tools, cloud services, and modern design culture to build scalable applications rapidly.

Reduce cost

By adopting the cloud-native approach, companies don't have to invest in the procurement and maintenance of costly physical infrastructure. This results in long-term savings in operational expenditure. The cost savings of building cloud-native solutions might also benefit your clients.

Ensure availability

Cloud-native technology allows companies to build resilient and highly available applications. Feature updates do not cause downtime and companies can scale up app resources during peak seasons to provide a positive customer experience.

What are cloud-native applications?

Cloud-native applications are software programs that consist of multiple small, interdependent services called microservices. Traditionally, developers built monolithic applications with a single block structure containing all the required functionalities. By using the cloud-native approach, software developers break the functionalities into smaller microservices. This makes cloud-native applications more agile as these microservices work independently and take minimal computing resources to run.

Cloud-native applications compared to traditional enterprise applications

Traditional enterprise applications were built using less flexible software development methods. Developers typically worked on a large batch of software functionalities before releasing them for testing. As such, traditional enterprise applications took longer to deploy and were not scalable.

On the other hand, cloud-native applications use a collaborative approach and are highly scalable on different platforms. Developers use software tools to heavily automate building, testing, and deploying procedures in cloud-native applications. You can set up, deploy, or duplicate microservices in an instant, an action that's not possible with traditional applications.

What is cloud-native application architecture?

The cloud-native architecture combines software components that development teams use to build and run scalable cloud-native applications. The CNCF lists immutable infrastructure, microservices, declarative APIs, containers, and service meshes as the technological blocks of cloud-native architecture.

Immutable infrastructure

Immutable infrastructure means that the servers for hosting cloud-native applications remain unchanged after deployment. If the application requires more computing resources, the old server is discarded, and the app is moved to a new high-performance server. By avoiding manual upgrades, immutable infrastructure makes cloud-native deployment a predictable process.

Microservices

Microservices are small, independent software components that collectively perform as complete cloud-native software. Each microservice focuses on a small, specific problem. Microservices are loosely coupled, which means that they are independent software components that communicate with each other. Developers make changes to the application by working on individual microservices. That way, the application

continues to function even if one microservice fails.

API

Application Programming Interface (API) is a method that two or more software programs use to exchange information. Cloud-native systems use APIs to bring the loosely coupled microservices together. API tells you what data the microservice wants and what results it can give you, instead of specifying the steps to achieve the outcome.

Service mesh

Service mesh is a software layer in the cloud infrastructure that manages the communication between multiple microservices. Developers use the service mesh to introduce additional functions without writing new code in the application.

Containers

Containers are the smallest compute unit in a cloud-native application. They are software components that pack the microservice code and other required files in cloud-native systems. By containerizing the microservices, cloud-native applications run independently of the underlying operating system and hardware. This means that software developers can deploy cloud-native applications on premises, on cloud infrastructure, or on hybrid clouds. Developers use containers for

packaging the microservices with their respective dependencies, such as the resource files, libraries, and scripts that the main application requires to run.

Benefits of containers

Some benefits of containers include:

- You use fewer computing resources than conventional application deployment
- You can deploy them almost instantly
- You can scale the cloud computing resources your application requires more efficiently

What is cloud-native application development?

Cloud-native application development describes how and where developers build and deploy cloud-native applications. A cultural shift is important for cloud-native development. Developers adopt specific software practices to decrease the software delivery timeline and deliver accurate features that meet changing user expectations. We give some common cloud-native development practices below.

Continuous integration

Continuous integration (CI) is a software practice in which developers integrate changes into a shared code base frequently and without errors. Small, frequent changes make development more efficient because you can identify and

troubleshoot issues faster. CI tools automatically assess the code quality for every change so that development teams can add new features with greater confidence.

Continuous delivery

Continuous delivery (CD) is a software practice that supports cloud-native development. With CD, development teams ensure that the microservices are always ready to be deployed to the cloud. They use software automation tools to reduce risk when making changes, such as introducing new features and fixing bugs on applications. CI and CD work together for efficient software delivery.

DevOps

DevOps is a software culture that improves the collaboration of development and operations teams. It is a design philosophy that aligns with the cloud-native model. DevOps practices allow organizations to speed up the software development lifecycle. Developers and operation engineers use DevOps tools to automate cloud-native development.

Serverless

Serverless computing is a cloud-native model where the cloud provider fully manages the underlying server infrastructure. Developers use serverless computing because the cloud infrastructure automatically scales and configures

to meet application requirements. Developers only pay for the resources the application uses. The serverless architecture automatically removes compute resources when the app stops running.

What are the benefits of cloud-native application development?

Faster development

Developers use the cloud-native approach to reduce development time and achieve better quality applications. Instead of relying on specific hardware infrastructure, developers build ready-to-deploy containerized applications with DevOps practices. This allows developers to respond to changes quickly. For example, they can make several daily updates without shutting down the app.

Platform independence

By building and deploying applications in the cloud, developers are assured of the consistency and reliability of the operating environment. They don't have to worry about hardware incompatibility because the cloud provider takes care of it.

Therefore, developers can focus on delivering values in the app instead of setting up the underlying infrastructure.

Cost-effective operations

You only pay for the resources your application actually uses. For example, if your user traffic

spikes only during certain times of the year, you pay additional charges only for that time period. You do not have to provision extra resources that sit idle for most of the year.

What is cloud-native stack?

Cloud-native stack describes the layers of cloud-native technologies that developers use to build, manage, and run cloud-native applications. They are categorized as follows.

Infrastructure layer

The infrastructure layer is the foundation of the cloud-native stack. It consists of operating systems, storage, network, and other computing resources managed by third-party cloud providers.

Provisioning layer

The provisioning layer consists of cloud services that allocate and configure the cloud environment.

Runtime layer

The runtime layer provides cloud-native technologies for containers to function. This comprises cloud data storage, networking capability, and a container runtime such as containerd.

Orchestration and management layer

Orchestration and management are responsible for integrating the various cloud components so that they function as a single unit. It is similar to how an operating system works in traditional computing. Developers use orchestration tools like Kubernetes to deploy, manage, and scale cloud applications on different machines.

Application definition and development layer

This cloud-native stack layer consists of software technologies for building cloud-native applications. For example, developers use cloud technologies like database, messaging, container images, and continuous integration (CI) and continuous delivery (CD) tools to build cloud applications.

Observability and analysis tools

Observability and analysis tools monitor, evaluate, and improve the system health of cloud applications. Developers use tools to monitor metrics like CPU usage, memory, and latency to ensure there is no disruption to the app's service quality.

What is cloud computing?

Cloud computing refers to software infrastructure hosted on an external data center and made available to users on a pay-per-use basis. Companies don't have to pay for expensive servers and maintain them. Instead, they can use

on-demand cloud-native services such as storage, database, and analytics from a cloud provider.

Cloud computing compared to cloud native

Cloud computing is the resources, infrastructure, and tools provided on-demand by cloud vendors.

Meanwhile, cloud native is an approach that builds and runs software programs with the cloud computing model.

What is cloud-enabled?

Cloud-enabled applications are legacy enterprise applications that were running on an on-premises data center but have been modified to run on the cloud. This involves changing part of the software module to migrate the application to cloud servers. You can thus use the application from a browser while retaining its original features.

Cloud native compared to cloud enabled

The term cloud native refers to an application that was designed to reside in the cloud from the start. Cloud native involves cloud technologies like microservices, container orchestrators, and auto scaling. A cloud-enabled application doesn't have the flexibility, resiliency, or scalability of its cloud-native counterpart. This is because cloud-enabled applications retain their monolithic structure even though they have moved to the cloud.

Why build cloud-native applications on AWS?

AWS offers the necessary technologies, tools, and services to develop functional cloud-native applications. You can focus on building software products instead of worrying about the underlying infrastructure:

- Move to managed [containers on AWS](#) to simplify operations, and reduce management overhead
- Build new applications or features using serverless technologies with [AWS Lambda](#) and purpose-built databases with [Amazon DynamoDB](#)
- Use tools like [AWS Amplify](#) and [AWS CDK](#) to maximize agility and accelerate development
- Choose from 15 relational and nonrelational purpose-built [AWS databases](#) to support microservices architecture and modern application needs, such as storing documents and key-value pairs
- Use our portfolio of [DevOps](#) services and our vast [Partner Network](#) to help develop and run applications faster and build applications at scale

Get started with cloud-native applications by creating an [AWS account](#) today.

Ending Support for Internet Explorer

[Got it](#)

AWS support for Internet Explorer ends on 07/31/2022. Supported browsers are Chrome, Firefox, Edge, and Safari. [Learn more »](#)

