ISBN: 978-81-978152-4-9

7. Cloud Native Application

Neha Bape

Assistant Professor, IT Department,
S.M.T. Jankibai college of Arts, Science & Commerce,
Kalwa, Thane Maharashtra.

7.1 What Is Cloud Native:

The software method known as "cloud native" is used to develop, implement, and oversee contemporary applications in cloud computing settings. The goal of modern businesses is to create apps that are incredibly durable, scalable, and adaptable so they can easily upgrade them to suit changing client needs. They accomplish this by utilizing cutting-edge methods and instruments that naturally facilitate application development on cloud infrastructure. Adopters gain a creative, competitive edge from these cloud-native technologies, which enable quick and frequent updates to apps without affecting service delivery.

Cloud-native applications: what are they?

Applications that are cloud-native are composed of several tiny, interconnected services known as microservices. In the past, programmers created monolithic apps with a single block structure holding all necessary features. Software engineers divide up the capabilities into smaller microservices by employing the cloud-native methodology. Because these microservices are self-contained and require little processing power to operate, they increase the agility of cloud-native apps.

Cloud-native applications compared to traditional enterprise applications:

Less flexible software development techniques were used in the creation of traditional enterprise apps. Before making a significant batch of software

functionalities available for testing, developers usually worked on them. Because of this, deploying traditional enterprise apps was slower and they were not scalable.

Conversely, cloud-native applications are extremely scalable across various platforms and employ a collaborative methodology. In cloud-native apps, developers employ software tools to greatly automate the processes of building, testing, and deploying applications. Microservices allow for fast setup, deployment, and duplication of operations that are not feasible with traditional applications.

7.1.1 What is the CNCF?

An open-source group called the Cloud Native Computing group (CNCF) assists businesses in launching their cloud-native initiatives. The CNCF was founded in 2015 with the goal of assisting the open-source community in creating essential cloud-native components, such as Kubernetes. Amazon is a CNCF member.

What Is Cloud-Native Application Architecture?

Development teams can create and execute scalable cloud-native applications by combining software components found in the cloud-native architecture. The building pieces of cloud-native architecture are listed by the CNCF as immutable infrastructure, microservices, declarative APIs, containers, and service meshes.

Unchangeable infrastructure:

When cloud-native apps are hosted on servers with immutable infrastructure, the servers stay the same after they are deployed. The old server is deleted and the program is relocated to a new high-performance server if it needs additional processing power. Immutable infrastructure makes cloud-native deployment predictable by eliminating the need for manual upgrades.

Small-Scale Services:

Micro services are discrete, standalone software elements that work together to construct full cloud-native applications. Every microservice concentrates on a single, tiny issue. Microservices are autonomous software components that interact with one another because they are loosely connected. Developers work on individual micro services to make modifications to the application. In this manner, in the event that one micro service fails, the program will still work.

API:

Information sharing between two or more software programs can be done using the Application Programming Interface (API). These loosely coupled microservices are connected by APIs in cloud-native applications. Instead of outlining how to get there, the API tells you what information the microservice needs and what kinds of outputs it can produce.

Mesh services:

A software layer in the cloud infrastructure called service mesh controls how many microservices communicate with one another. By using the service mesh, programmers can add new features to their applications without having to update the code.

7.2 Containers:

The smallest computational unit in a cloud-native application is called a container. In cloud-native systems, they are software components that bundle the microservice code and other necessary files.

Cloud-native apps operate independently of the underlying hardware and operating system thanks to the containerization of the microservices.

This implies that cloud-native apps can be implemented by software developers on-site, on cloud infrastructure, or on hybrid clouds. Containers are used by developers to package microservices along with their necessary dependencies, which include the libraries, scripts, and resource files needed for the primary application to function.

7.2.1 The Advantages of Containers:

Among the advantages of containers are:

1. Utilizing less processing power than traditional application deployment
2. They can be used practically immediately.
3. Scaling the cloud computing resources your application needs can be done more effectively.

What is the creation of cloud-native applications?

The process and locations by which developers create and implement cloud-native apps are outlined in cloud-native application development. A shift in mindset is necessary for cloud-native programming. In order to reduce the time required for software delivery and provide precise features that satisfy evolving customer expectations, developers implement particular software methods. Below are a few standard cloud-native development techniques.

Continuous Integration:

Developers that use continuous integration (CI) routinely and error-free incorporate changes into a common code base. Development is more efficient when small, frequent modifications are made since problems may be found and fixed more quickly. Development teams can add new features with more confidence since continuous integration (CI) systems automatically evaluate the quality of the code for each change.

DevOps:

DevOps is a software culture that enhances the cooperation between teams working on development and operations. It is a cloud-native model-aligned design philosophy. Organizations can accelerate the software development lifecycle by implementing DevOps methods. DevOps tools are used by developers and operation engineers to automate cloud-native development**.**

Serverless Computing:

A cloud-native approach known as "serverless computing" gives the cloud provider complete control over the underlying server infrastructure. Because cloud infrastructure automatically expands and configures to match application requirements, serverless computing is popular among developers. Only the resources used by the program are paid for by the developers. When an application terminates, the serverless architecture immediately releases the computational resources

7.2.2 What Are the Benefits of Cloud-Native Application Development?

Quicker growth:

The cloud-native approach is used by developers to produce applications of higher quality in less time. Using DevOps techniques, developers create ready-to-deploy containerized apps rather of depending on particular hardware infrastructure. This enables developers to react swiftly to modifications. They can update the software multiple times a day, for instance, without having to close it.

Platform autonomy:

Developers may be confident in the consistency and dependability of the operating environment when they create and implement apps in cloud environments.

The cloud provider takes care of hardware mismatch, so they don't need to worry about it. As a result, rather of concentrating on establishing the underlying infrastructure, developers can concentrate on providing value in the app.

Economical operations:

Only the resources that your program really utilizes are charged for. As an illustration, if your user traffic

7.3 An Example Cloud-Native Applications:

A web-based e-commerce platform, such an online clothes store, would be a basic example of a cloud native application.

The program would be packed as a collection of containers, each of which would have a distinct purpose, such as the payment service, database, and web front-end. This process is known as containerization.

Each part of the program would be a separate service that interacts with other services via APIs because it would use a microservices design.

The application would deploy and manage the cloud-based containers using automation tools like Kubernetes.

A continuous integration and delivery (CI/CD) pipeline would be used by the application, enabling more frequent and quicker releases.

With characteristics like load balancing, auto-scaling, and self-healing, the application would be built to withstand failures.

In this scenario, the cloud environment's scalability, availability, and affordability would be utilized by the e-commerce platform to manage a high volume of users and transactions.

Furthermore, deployment and management are made simple by the use of containers and micro services, while development and deployment are accelerated by the use of automation tools and CI/CD pipeline.

7.4 Last Observation:

All things considered, cloud native applications provide a novel approach to developing and implementing software that fully utilizes the cloud computing paradigm. They are perfect for operating in a cloud environment because of their high levels of scalability, availability, and resilience to failures.