Cloud Native Apps:

Cloud Native application development is an approach to designing, building, and running complex and distributed systems. The approach takes full advantage of modern software development practices, tools, technologies, and cloud infrastructure. Cloud native changes the way you design, implement, deploy, and operationalize systems. We need to use Continuous Integration, container engines and orchestration.

Advantages of Cloud Native applications are improved speed, scalability, and product in cost effective way.

What is Cloud Native?

Cloud-native architecture and technologies are an approach to designing, constructing, and operating workloads that are built in the cloud and take full advantage of the cloud computing model.

The [Cloud Native Computing Foundation](https://www.cncf.io/) provides the [official definition](https://github.com/cncf/foundation/blob/master/charter.md):

*Cloud-native technologies empower organizations to build and run scalable applications in modern, dynamic environments such as public, private, and hybrid clouds. Containers, service meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.*

*These techniques enable loosely coupled systems that are resilient, manageable, and observable. Combined with robust automation, they allow engineers to make high-impact changes frequently and predictably with minimal toil.*

We get improved speed, scalable and productivity in cost effective way.

Following are the pillars of Cloud Native

1. The Cloud: Cloud Native systems takes full advantage of the cloud service model
2. Modern Design: The Twelve-Factor application is the widely accepted and used methodology for constructing cloud-based applications
3. Microservices: Cloud-Native systems embrace microservices, a popular architectural style for constructing modern application. Build small set of independent services that interact with the shared fabrics. Microservices provides agility. Microservices can be created upon any modern development platforms
4. Containers: Containers are a great enabler of cloud native software. Containerizing a microservice is simple and straightforward. The code, dependencies, runtime are packaged into binary called as container image. Containers provides portability and guarantee consistency across environments. By encapsulating everything into a single package, we are isolating the code and its dependencies from the underling infrastructure. To manage these containers, we have special software called container orchestrators
5. Backing Services: Cloud Native systems depends upon many different ancillary resources such as data stores, message brokers, monitoring and identity services etc. these services are called backing services. The best practice is to treat a backing service as attached resource, dynamically bound to a microservice with configuration information stored in external configuration.
6. Automation: With infrastructure as code, we can automate the platform provisioning and application deployments. We add versioning of code base, testing to DevOps practices. Our infrastructure and deployments are automated consistent and repeatable. We can use same scripts over and over without any side effects, only the updated resources are affected.

What is multi-cloud?

Multi-Cloud is the superset of multiple public cloud, hybrid, on-premises, and edge. A multi-cloud deployment model relies on the use of more than one public cloud service provider for compute or storage resources, independent of the use of other private cloud or on-premises infrastructure. A multi-cloud deployment that includes private cloud or on-premises infrastructure is considered a hybrid multi-cloud.

Why Multi-Cloud?

1. Multi-cloud use cases can be leveraged to offer IT teams increases flexibility and control over workloads and data.
2. As multi-cloud offers flexible cloud environment, organizations can meet specific workload or application requirements- both technically and commercially- by adopting it.
3. Organizations also see geographic advantages to using multiple cloud providers, to address app latency problems.
4. We may start using specific cloud providers for short time to achieve short-term goals and then stop using it.

Multi-cloud environments come with their challenges—complexity, resources, expertise, cost, and management issues. Here are 5 things to keep in mind when building an effective multi-cloud environment.

1. Analyze entire network, and then identify which service of cloud provider is the best for specific requirement, to avoid system complexity and poor utilization of resources.
2. Having multiple cloud providers increases low-level maintenance and monitoring tasks. It’s better to automate them
3. Focus on policy standardization that are applied automatically to each cloud environment. The policies cover such areas as data storage, workloads, traffic flows, virtual servers, compliance/ regulations, security, and reporting.
4. Use integrated data center management system designed for virtual environments. It helps building system in which server, network, storage, operations, security, and applications teams work according to common goals.
5. Identify the apps that are best suited for multi-cloud environment. Unlike traditional apps, cloud-native apps are flexible and service-oriented, comprised of collections of containers and services, based on a scale-out architecture. Also, these are easy to automate, move, and scale