22AIE305: CLOUD COMPUTING

What is a cloud-native application?

- Cloud-native applications (CNA) are purposely built to take advantage of the cloud elasticity and scalability and multiple cloud services (messaging, service discovery, AI, VR, IoT, data science, and more) offered by cloud providers.
- 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 application development describes how and where developers build and deploy CNA.
- A cultural shift is important for CNA development.
- Developers adopt specific software practices to decrease software delivery timeline and deliver accurate features that meet changing user expectations.
- Cloud-native organizations use automation to reduce human error, make repeatable processes more efficient, and take fine-grained control over their application infrastructure.

CNA vs Traditional 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 are 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 micro-services in an instant, an action that's not possible with traditional applications.

Why build cloud-native applications?

- The cloud-native approach to software development helps:
- Speed up application development and updates to **achieve faster time to market**.
- **Keep an application highly available** even in case of overloads or a significant increase in the number and complexity of user requests.
- **Reduce development costs** due to the use of cloud services.
- **Save significant operation costs** through paying only for the time cloud services and resources are rented.
- Accelerate the business' **ability to innovate** due to easy access to high-performance processors, analytics services, AI services.

Benefits of CNA 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.

Benefits of CNA development cont'd..

- **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.

Benefits of CNA development cont'd..

- **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.

Key Principles of Cloud-Native Development

- Microservices Architecture: Applications are broken down into small, loosely coupled services that communicate via APIs. Each service can be independently developed, deployed, and scaled.
- Containerization: Applications and services are packaged into containers (e.g., using Docker) to ensure they run consistently across different environments (development, testing, and production).
- DevOps and Continuous Delivery: Cloud-native applications often follow CI/CD pipelines to automate building, testing, and deploying. DevOps tools like Jenkins, CircleCI, or GitHub Actions are common.
- Resilience and Scalability: Cloud-native apps are built to handle failures gracefully (resilient to downtime) and can automatically scale based on demand.
- Infrastructure as Code (IaC): Infrastructure provisioning and management is automated using tools like Terraform, AWS CloudFormation, or Kubernetes.
- API-First Design: Services communicate primarily through APIs (often RESTful or GraphQL), making them more flexible and decoupled.

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.

Containers

- Containers are the smallest compute unit in a CNA. 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 OS and hardware.
- This means that 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

- 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

Containerization

- containerization is a form of virtualization where applications are configured to run in isolated environments, called *containers*, while still using the host OS' kernel.
- Containers are much more lightweight than traditional virtualization.
- They rely on the host OS to do much of the heavy lifting, rather than a hypervisor or other VM layer.

Advantages of Containerization

- The advantage of containerization in a cloud-native environment is that containers are lightweight, portable and -- most importantly -- repeatable.
- This enables developers to write, test and deploy applications in consistent environments while minimizing cost.

Splitting an application in microservices and getting it containerized

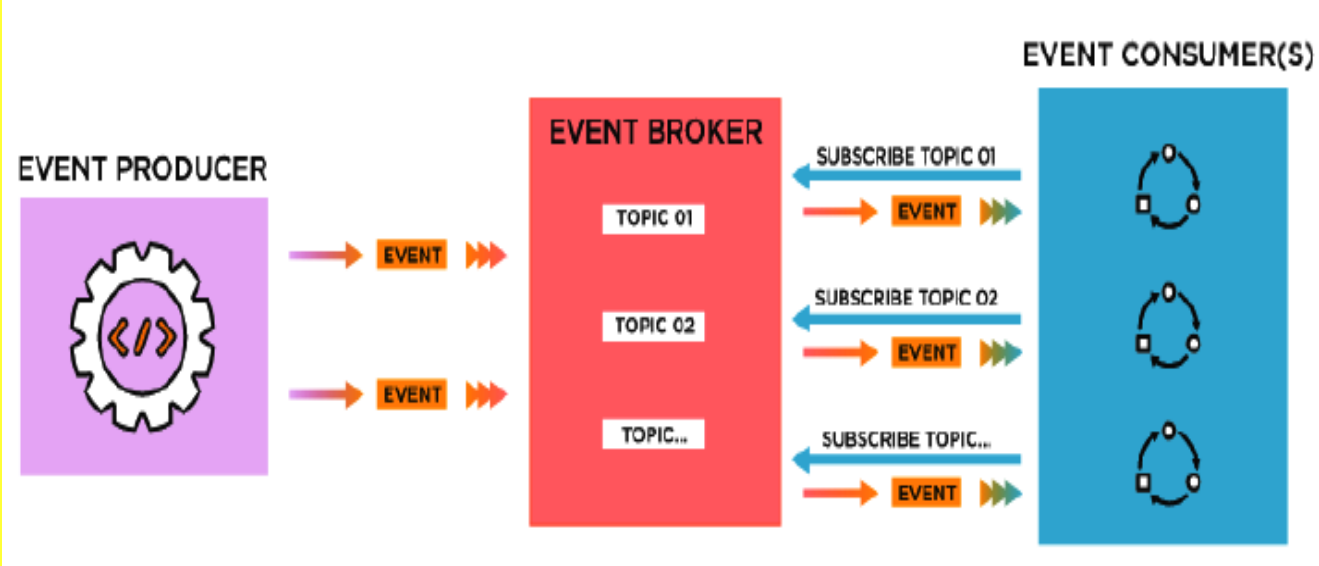
- For an app to benefit from cloud scalability, easy portability and high resilience, it should be subdivided into small isolated components. Thus, we design cloud-native applications as a set of loosely coupled, purpose-centered microservices that exist and operate independently and communicate with each other via lightweight APIs. You can see the benefits of the microservices-based architecture on the example of ScienceSoft's project on hotel self-service app development.
- I also recommend packaging each service into containers with the required operating system and dependency libraries. The use of containers helps reduce the complexity of distributed application management and to further simplify its scalability and portability.

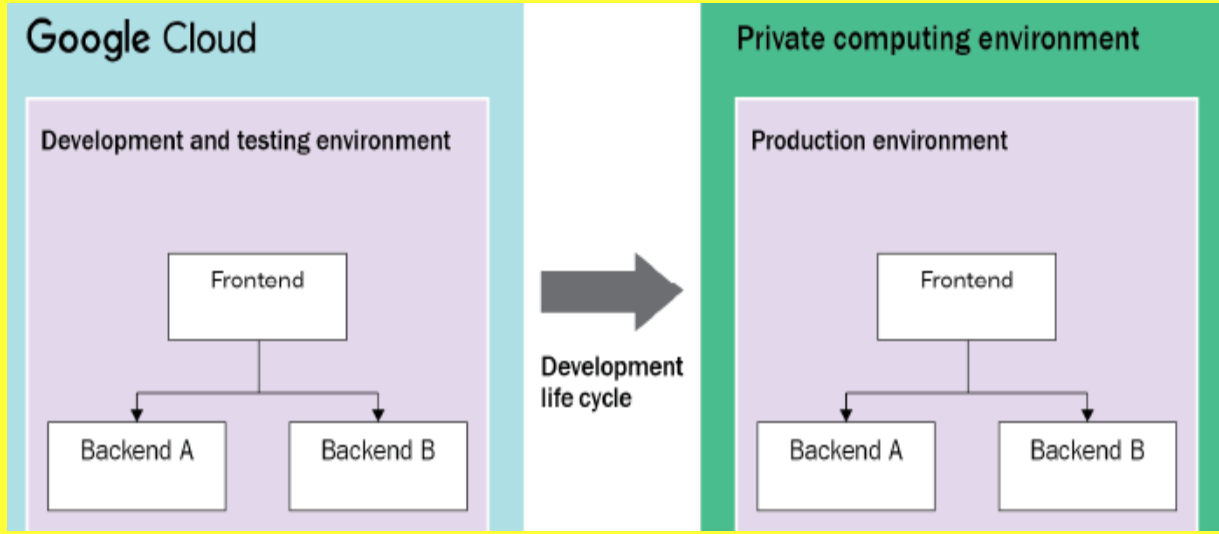
Orchestration

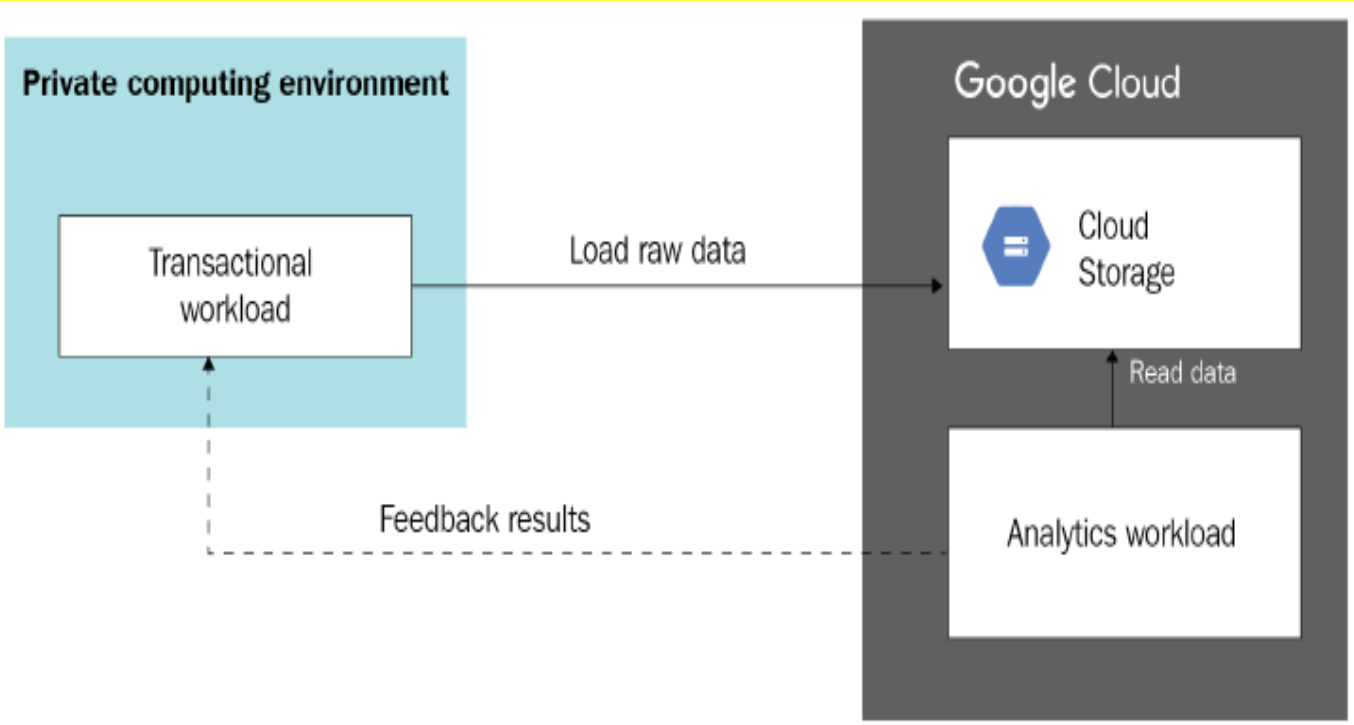
- Orchestration is the process of automating the lifecycle management of all containers in a production environment to efficiently manage tasks like container creation and destruction, horizontal scaling, version control, and networking and dependency management.
- Kubernetes is one of the most popular tools for container orchestration. The more lightweight Docker Compose is often used for local orchestration. Regardless of the tool, successful management of production resources always require some sort of orchestration.
- The application components are containerized and the containers are orchestrated by a central orchestrator.

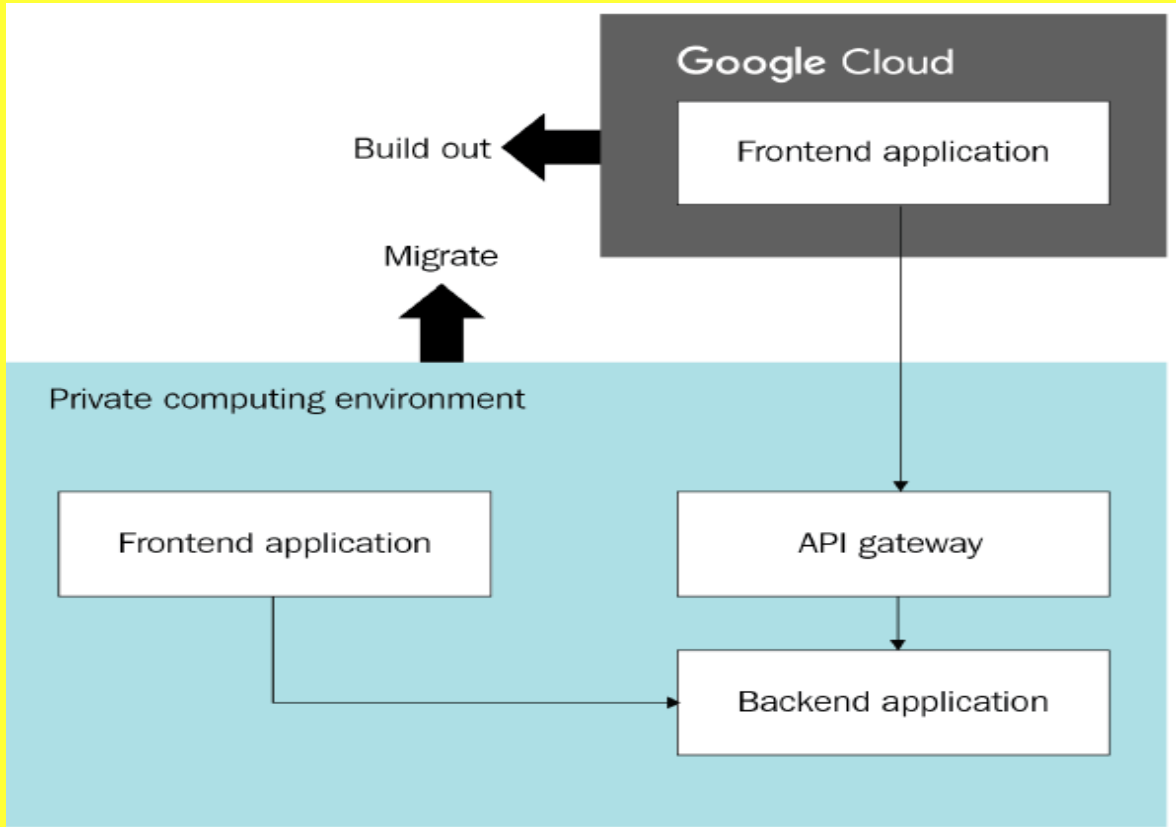
SOA

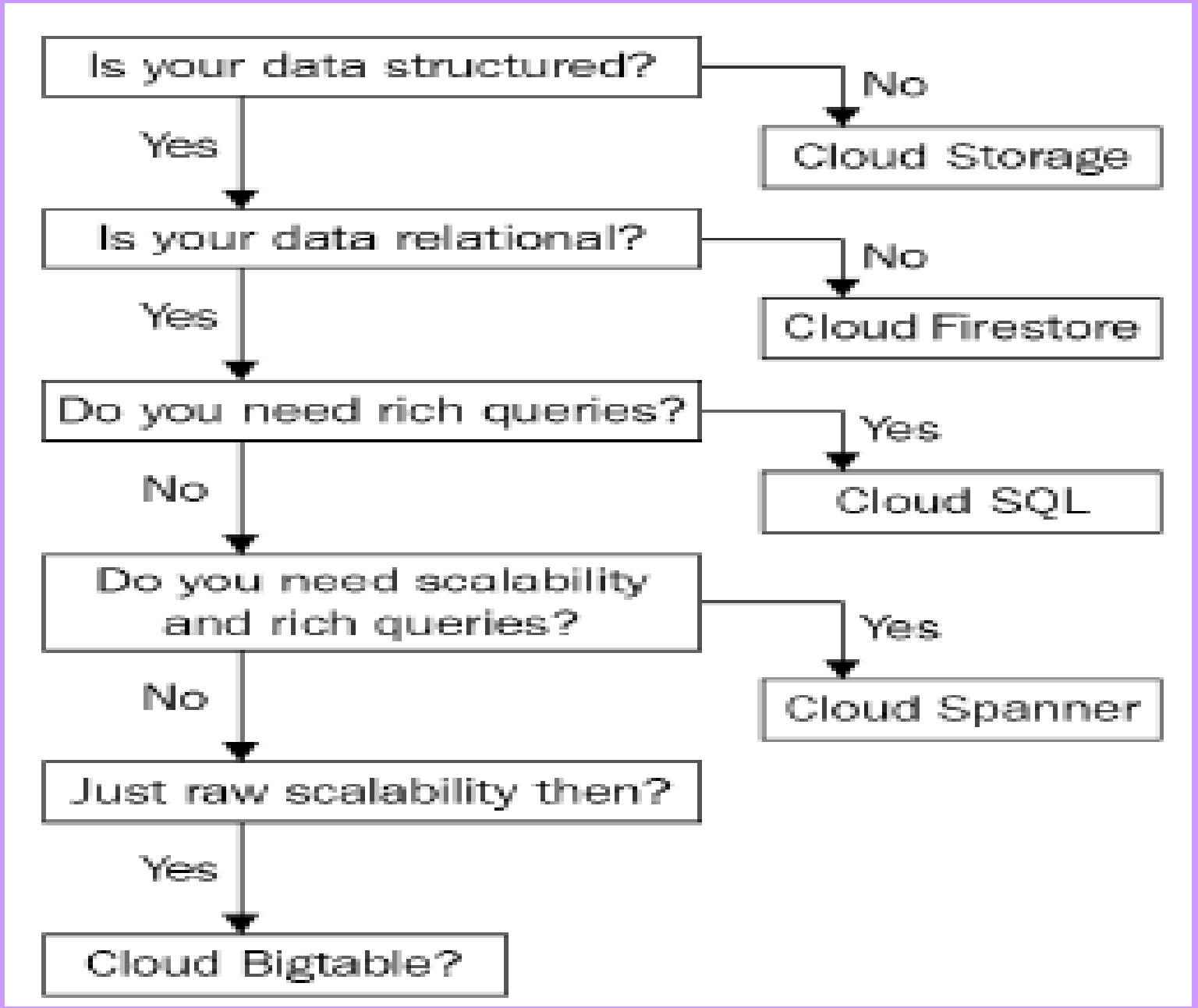
- Service Oriented Architecture (SOA) describes a standard method for requesting services from distributed components or the Cloud, and managing the results.
- SOA Clients and components can be written in different programming languages and can use multiple messaging and networking protocols to communicate with each other.


EVENT PRODUCER
EVENT
EVENT
EVENT BROKER
TOPIC 01
TOPIC 02
TOPIC...
SUBSCRIBE TOPIC 01
EVENT
SUBSCRIBE TOPIC 02
EVENT
SUBSCRIBE TOPIC...
EVENT
EVENT CONSUMER(S)


Google Cloud
Development and testing environment
Frontend
Backend A
Backend B
Development life cycle
Private computing environment
Production environment
Frontend
Backend A
Backend B


Private computing environment
Transactional workload
Load raw data
Feedback results
Google Cloud
Cloud Storage
Read data
Analytics workload


Google Cloud
Frontend application
Build out
Migrate
Private computing environment
Frontend application
API gateway
Backend application


Is your data structured?
No
Cloud Storage
Yes
Is your data relational?
No
Cloud Firestore
Yes
Do you need rich queries?
Yes
Cloud SQL
No
Do you need scalability and rich queries?
Yes
Cloud Spanner
No
Just raw scalability then?
Yes
Cloud Bigtable?

What is the CNCF?

- Cloud-native application development with JavaScript involves creating applications designed to leverage cloud computing architectures, platforms, and services.
- What is the CNCF?
- The Cloud Native Computing Foundation (CNCF) is an open-source foundation that helps organizations kick start their cloud-native journey. Established in 2015, the CNCF supports the open-source community in developing critical cloud-native components, including Kubernetes. Amazon is a member of CNCF.

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.

Application Programming Interface (API)

- 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.

Layers of CNA technology

- **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.

Layers of CNA technology cont'd

- **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.

Automation

- Automation is essential to complex distributed systems for rapid development and continuous improvement with frequent releases. As manual efforts are too slow and error-prone to guarantee that, robust automation for a CNA helps keep the app well-tested, secure, reliable and scalable.
- While it is possible to manually build, test, deploy and manage your application infrastructure, cloud providers offer tools to abstract away many of those manual processes.
- This enables cloud-native organizations to be more efficient in everything from their hiring to their infrastructure needs.