Core Cloud Services – Cloud Compute Options

Essential Cloud Compute Concepts

What is Azure compute?

Cloud compute is an on-demand computing service for running cloud-based applications. It provides computing resources like multi-core processors and supercomputers via virtual machines and containers. It also provides serverless computing to run apps without requiring infrastructure setup or configuration. The resources are available on-demand and can typically be created in minutes or even seconds. You pay only for the resources you use and only for as long as you're using them.

There are four common techniques for performing compute in Azure:

- Virtual machines
- Containers
- Azure App Service
- Serverless computing

What are virtual machines?

Virtual machines, or VMs, are software emulations of physical computers. They include a virtual processor, memory, storage, and networking resources. They host an operating system (OS), and you're able to install and run software just like a physical computer. And by using a remote desktop client, you can use and control the virtual machine as if you were sitting in front of it.

What are containers?

Containers are a virtualization environment for running applications. Just like virtual machines, containers run on top of a host operating system. But unlike VMs, containers don't include an operating system for the apps running *inside* the container. Instead,

containers bundle the libraries and components needed to run the application and use the existing host OS running the container. For example, if five containers are running on a server with a specific Linux kernel, all five containers and the apps within them share that same Linux kernel.

What is Azure App Service?

Azure App Service is a platform-as-a-service (PaaS) offering in Azure that is designed to host enterprise-grade web-oriented applications. You can meet rigorous performance, scalability, security, and compliance requirements while using a fully managed platform to perform infrastructure maintenance.

What is Serverless Computing?

Serverless computing is a cloud-hosted execution environment that runs your code but completely abstracts the underlying hosting environment. You create an instance of the service, and you add your code; no infrastructure configuration or maintenance is required, or even allowed.

Explore Containers

If you wish to run multiple instances of an application on a single host machine, containers are an excellent choice. The container orchestrator can start, stop, and scale out application instances as needed.

A container is a modified runtime environment built on top of a host OS that executes your application. A container doesn't use virtualization, so it doesn't waste resources simulating virtual hardware with a redundant OS. This environment typically makes containers more lightweight than VMs. This design allows you to respond quickly to changes in demand or failure. Another benefit of containers is you can run multiple isolated applications on a single container host. Since containers are secured and isolated, you don't need separate servers for each app.

Explore Serverless Computing

Serverless computing is the abstraction of servers, infrastructure, and OSs.
With *serverless* computing, Azure takes care of managing the server infrastructure and allocation/deallocation of resources based on demand. Infrastructure isn't your responsibility. Scaling and performance are handled automatically, and you are billed only for the exact resources you use. There's no need to even reserve capacity.

Serverless computing encompasses three ideas: the abstraction of servers, an event-driven scale, and micro-billing:

1. **Abstraction of servers**: Serverless computing abstracts the servers you run on. You never explicitly reserve server instances; the platform manages that for you. Each function execution can run on a different compute instance, and this execution context is transparent to the code. With serverless architecture, you simply deploy your code, which then runs with high availability.
2. **Event-driven scale**: Serverless computing is an excellent fit for workloads that respond to incoming events. Events include triggers by timers (for example, if a function needs to run every day at 10:00 AM UTC), HTTP (API and webhook scenarios), queues (for example, with order processing), and much more. Instead of writing an entire application, the developer authors a function, which contains both code and metadata about its triggers and bindings. The platform automatically schedules the function to run and scales the number of compute instances based on the rate of incoming events. Triggers define how a function is invoked and bindings provide a declarative way to connect to services from within the code.
3. **Micro-billing**: Traditional computing has the notion of per-second billing, but often, that's not as useful as it seems. Even if a customer's website gets only one hit a day, they still pay for a full day's worth of availability. With serverless computing, they pay only for the time their code runs. If no active function executions occur, they're not charged. For example, if the code runs once a day for two minutes, they're charged for one execution and two minutes of computing time.