“The Cloud”,

More than you want to Know, But less than you need to Know.

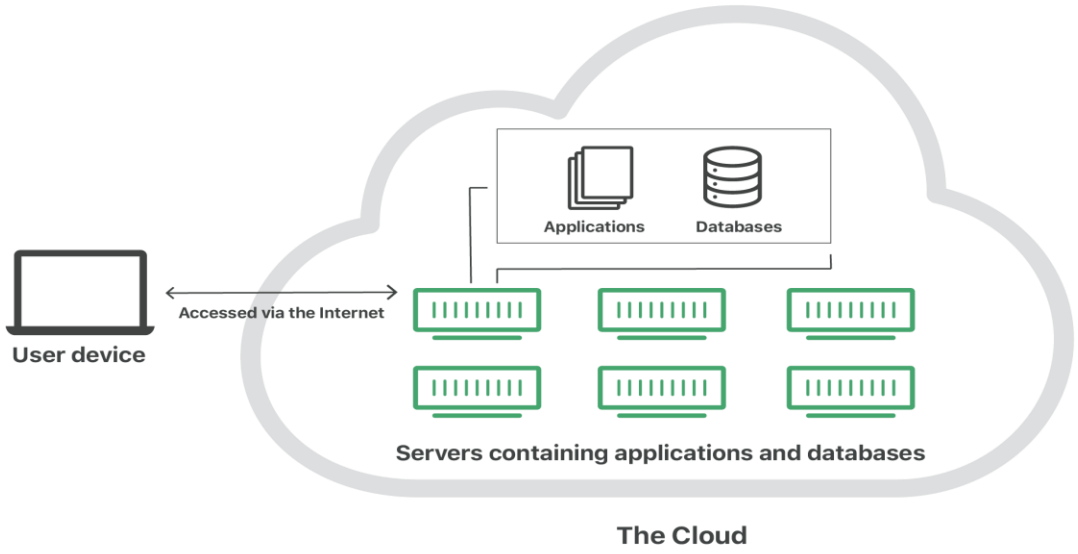
Why?

“The Cloud” is beginning to cover much of the space occupied by city planning. Knowing, understanding and using “The Cloud” as a tool and a resource is becoming more important. While the following presentation of elementary information and descriptions is both basic and incomplete, it is intended as a starter for a city planning and city administration thought process that will need to become much more sophisticated. The sources have been directly quoted with links to their sites.

City planners may or may not need to know the construction or technicalities of “The Cloud” but they darn sure need to know how to take advantage of its strengths and take care with its shortcomings. Fasten your parachute…

From…Cloudflare.

"The Cloud" refers to servers that are accessed over the Internet, and the software and databases that run on those servers. Cloud servers are located in data centers all over the world. By using cloud computing, users and companies don't have to manage physical servers themselves or run software applications on their own machines.



What are the main service models of cloud computing?

Software-as-a-Service (SaaS): Instead of users installing an application on their device, SaaS applications are hosted on cloud servers, and users access them over the Internet. SaaS is like renting a house: the landlord maintains the house, but the tenant mostly gets to use it as if they owned it. Examples of SaaS applications include Salesforce, MailChimp, and Slack.

Platform-as-a-Service (PaaS): In this model, companies don't pay for hosted applications; instead they pay for the things they need to build their own applications. PaaS vendors offer everything necessary for building an application, including development tools, infrastructure, and operating systems, over the Internet. PaaS can be compared to renting all the tools and equipment necessary for building a house, instead of renting the house itself. PaaS examples include Heroku and Microsoft Azure.

Infrastructure-as-a-Service (IaaS): In this model, a company rents the servers and storage they need from a cloud provider. They then use that cloud infrastructure to build their applications. IaaS is like a company leasing a plot of land on which they can build whatever they want – but they need to provide their own building equipment and materials. IaaS providers include DigitalOcean, Google Compute Engine, and OpenStack.

What about containers? Are containers IaaS, PaaS, SaaS, or FaaS?

Like virtual machines, containers are a cloud virtualization technology. They are part of the PaaS (Platform-as-a-Service) cloud model. Virtualization for containers occurs one abstraction layer up from where it occurs for virtual machines, at the operating system level instead of at the kernel level (the kernel is the foundation of the operating system, and it interacts with the computer's hardware). Each virtual machine has its own operating system kernel, but containers on the same machine share the same kernel.

Link: https://www.cloudflare.com/learning/cloud/what-is-the-cloud/

From…Microsoft.

Cloud computing is renting resources, like storage space or CPU cycles, on another company's computers. You only pay for what you use. The company providing these services is referred to as a cloud provider. Some example providers are Microsoft, Amazon, and Google.

The cloud provider is responsible for the physical hardware required to execute your work, and for keeping it up-to-date. The computing services offered tend to vary by cloud provider. However, typically they include:

- **Compute power -** such as Linux servers or web applications used for computation and processing tasks.
- **Storage -** such as files and databases.
- **Networking -** such as secure connections between the cloud provider and your company.
- **Analytics -** such as visualizing telemetry and performance data.

Containers provide a consistent, isolated execution environment for applications. They're similar to VMs [virtual machines] except they don't require a guest operating system. Instead, the application and all its dependencies are packaged into a "container" and then a standard runtime environment is used to execute the app. This allows the container to start up in just a few seconds, because there's no OS to boot and initialize. You only need the app to launch.

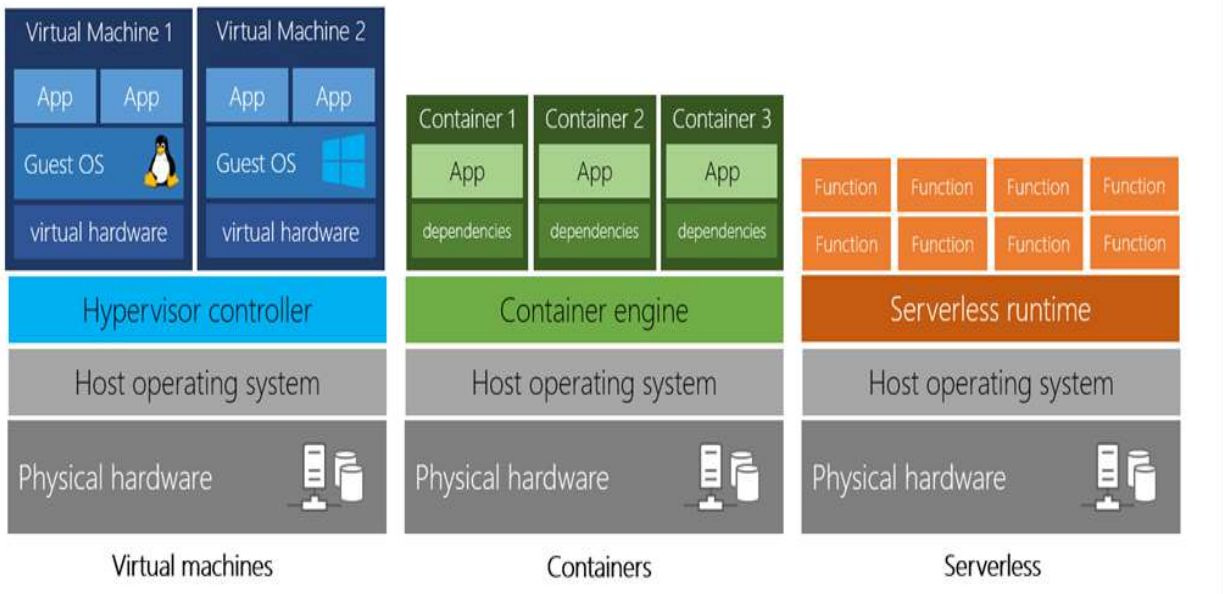
The open-source project, Docker, is one of the leading platforms for managing containers. Docker containers provide an efficient, lightweight approach to application deployment because they allow different components of the application to be deployed independently into different containers. Multiple containers can be run on a single machine, and containers can be moved between machines. The portability of the container makes it easy for applications to be deployed in multiple environments, either on-premises or in the cloud, often with no changes to the application.

What is serverless computing?

Serverless computing lets you run application code without creating, configuring, or maintaining a server. The core idea is that your application is broken into separate *functions* that run when triggered by some action. This is ideal for automated tasks - for example, you can build a serverless process that automatically sends an email confirmation after a customer makes an online purchase.

The serverless model differs from VMs and containers in that you only pay for the processing time used by each function as it executes. VMs and containers are charged while they're running - even if the applications on them are idle. This architecture doesn't work for every app - but when the app logic can be separated to independent units, you can test them separately, update them separately, and launch them in microseconds, making this approach the fastest option for deployment.

Here's a diagram comparing the three compute approaches we've covered.



Copy

The three verticals, virtual machines, containers, and serverless, show different architectures. Virtual machines start at physical hardware and has layers built on it: host operating system, hypervisor controller, and then two virtual machines on top with one running Linux and two apps and one running Windows and two apps. Containers starts with physical hardware with additional layers: host operating system, container engine, and then three containers, each with their own dependencies and hosted apps. Serverless starts with physical hardware with additional layers: host operating system, serverless runtime, and then eight functions.

Storage

Most devices and applications read and/or write data. Here are some examples:

- Buying a movie ticket online
- Looking up the price of an online item
- Taking a picture
- Sending an email
- Leaving a voicemail

In all of these cases, data is either *read* (looking up a price) or *written* (taking a picture). The type of data and how it's stored can be different in each of these cases.
Cloud providers typically offer services that can handle all of these types of data. For example, if you wanted to store text or a movie clip, you could use a file on disk. If you had a set of relationships such as an address book, you could take a more structured approach like using a database.

The advantage to using cloud-based data storage is you can scale to meet your needs. If you find that you need more space to store your movie clips, you can pay a little more and add to your available space. In some cases, the storage can even expand and contract automatically - so you pay for exactly what you need at any given point in time.
Link: https://docs.microsoft.com/en-us/learn/modules/principles-cloud-computing/2-what-is-cloud-computing

BANK INFO SECURITY®

From…

The six principles - detailed in the recently published ISACA publication **Guiding Principles for Cloud Computing Adoption and Use** - include enablement, cost/benefit, enterprise risk, capability, accountability and trust. Here's how ISACA defines each of those principles:

1. **Enablement:** Plan for cloud computing as a strategic enabler, rather than as an outsourcing arrangement or technical platform.
2. **Cost/benefit:** Evaluate the benefits of cloud acquisition based on a full understanding of the costs of cloud compared with the costs of other technology platform business solutions.

3. **Enterprise risk:** Take an enterprise risk management perspective to manage the adoption and use of cloud.
4. **Capability:** Integrate the full extent of capabilities that cloud providers offer with internal resources to provide a comprehensive technical support and delivery solution.
5. **Accountability:** Manage accountabilities by clearly defining internal and provider responsibilities.
6. **Trust:** Make trust an essential part of cloud solutions, building trust into all business processes that depend on cloud computing.

The cloud's availability means the technology infrastructure is not the market differentiator it has been in the past.

LINK: https://www.bankinfosecurity.com/blogs/6-principles-for-effective-cloud-computing-p-1217

From…Stanford University IT.

Principles

The goal of this document is to summarize some of the more important aspects of running IT infrastructure, applications, and related services as cloud deployments. However, while many of these design principles and patterns are not particular to the cloud, and could be applied locally, they become necessary when building reliable cloud services. While it is possible to build cloud-based systems the way we have traditionally, some of our local best practices are in conflict with cloud best practices. For example, PXE-booting new servers (physical or virtual), with manual sysadmin intervention to acquire Kerberos keytabs during the initial install doesn't work with cloud providers who provide no interactive console access (AWS EC2, for instance). Similarly, sizing a pool of servers to meet peak demand during a few days each year, rather than auto-scaling when needed, is a cloud anti-pattern.

- **Assume Campus Systems are not Available**
- **Servers Are Ephemeral**
  Cloud-based servers (aka "instances") are far more ephemeral; they are started when needed, with random IP addresses and DNS names, and when they are terminated, or die, they are often gone, leaving no trace.
- **Servers Are Stateless**
  Since cloud servers - and their associated local storage - are ephemeral, they should also be stateless. Data must be stored on external storage services; configuration data can be injected at startup, or stored in external data sources.
- **Always Use Auto Scaling**
  Due to the ephemeral nature of cloud servers, guaranteeing uptime would be troublesome if cloud providers did not provide autoscaling. While most people think autoscaling is only useful for high-traffic sites, to grow and shrink a pool of servers behind a load-balancer as traffic changes, it can also be used to ensure a minimum number of servers are always running.
  For example, a common practice in AWS is to set the minimum and maximum number of servers in an auto-scaling group to 1: if that single server dies, the auto-scaling service will automatically replace it.

- **Leverage the Features of Our Tools**
  We often deploy new tools without leveraging the advantages they bring.
- **Always Use Custom Images**
- **Build Loosely Coupled Services**
  When the number of servers in a load-balanced pool can vary from hour to hour, or the IP address of a single-instance service can change from day to day, services must be loosely coupled. External-facing services should be load-balanced; back-end databases should be clustered (or use a highly-available database service like AWS' Relational Database Service (RDS); any middleware or application server tiers should also be load-balanced.
- **Nothing should rely on specific IP addresses**.
- **Hybrid Architectures Should Also Be Loosely Coupled**
- **No Patching or Updates on Running (Virtual) Systems**
  Physical servers must be patched according to MinSec requirements.
- **Cost Optimization (and Control)**
  Cloud providers often provide alerting facilities for when the monthly bill exceeds a threshold; configure alerts for your cloud accounts with appropriate thresholds, and with notifications going to mailing lists.
- **Design Systems for Fault-Tolerance**
  Reboot and machine failure are considered normal. The services should be designed to handle reboot caused by system patching, self-upgrade, machine replacement etc. By designing to account for subsystem failure, the service associated with the failure will not be affected - that is, the design is fault-tolerant.
- **Secure Bastion Hosts**
  Secure bastion hosts should enforce multi-factor authentication (e.g. SSH key **and** Duo, or Kerberos **and** Duo), or only allow access via physically secured credentials (e.g. SSH keys generated on a PIN- and touch- protected Yubikey). While bastion hosts must be used to access other servers within the protected network, those hosts must not store credentials for access to servers. Bastion hosts using SSH keys should not allow users to upload additional trusted keys; only keys installed by configuration management should be trusted.

LINK: https://uit.stanford.edu/cloud-transformation/iaas-architecture-principles

From…cloud computing services from Google.

LINK: https://cloud.google.com/blog/products/application-development/5-principles-for-cloud-native-architecture-what-it-is-and-how-to-master-it

Access control lists (ACLs).

Access control inherent in IaaS service - IaaS provider platforms include network access controls that provide segmentation between applications, and application tiers, that is normally provided by firewalls.