

CLOUD COMPUTING

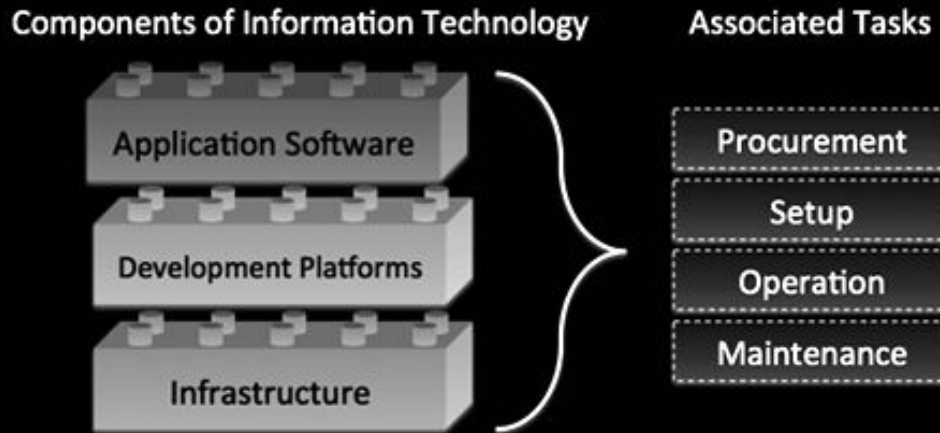
CE-5508



TECNOLOGÍAS DE INFORMACIÓN EN EL NEGOCIO

TI es esencial para los negocios actuales.

- Tradicionalmente TI tiene los siguientes componentes:



TECNOLOGÍAS DE INFORMACIÓN EN EL NEGOCIO

Implementar una solución de software implica más que solo codificar.

- Planear (*capacity-planning*), conseguir, configurar y mantener la infraestructura
- Mantener la plataforma de desarrollo de software y todo lo que implica
- Construir el software sobre dicha plataforma de software y hardware



El negocio tiene control completo sobre la solución de software desde cualquier perspectiva



Unlimited power!

SIEMPRE HAY UN PERO...



¿GRAN RESPONSABILIDAD?

PAGO POR ADELANTADO

Compromiso a largo plazo con un costo inicial de capital alto (CAPEX). Los recursos adquiridos sufren de depreciación y obsolescencia. Al ser activos, no son deducibles de los impuestos en un solo tracto, sino que se deducen parcialmente año tras año.

RESPONSABILIDAD TOTAL

Contratos de mantenimiento de Hardware.
Contratación de personal especializado.
Aprovisionamiento de soluciones eléctricas y enfriamiento (gastos recurrentes).

ESCALABILIDAD

Comprar más hardware o mejorar el existente para hacer frente a los picos de demanda (¿qué hacemos cuando la demanda es baja?)

BAJA UTILIZACIÓN

Hardware poderoso que pasa ocioso una gran parte del tiempo.

“¿Por qué tengo que gastar dinero
en IT si lo que vendo son zapatos?”

EL NEGOCIO



IT COMO SERVICIO

Transformar IT de producto a servicio

- Este tipo de transformaciones no son nuevas
- Pagar sólo por lo que se usa, cuando se usa, de la misma forma que una utilidad o servicio más.
- Aprovisionamiento rápido de recursos, lo que permite reducción del time-to-market
- Escalabilidad rápida
- Mejor utilización de recursos y reducción de huella de carbono



AGUA



GAS



ELECTRICIDAD

LOS PROVEEDORES DE SERVICIO

Para un proveedor de servicio, el servicio como tal es su negocio

- Especializados en todos los detalles referentes a la venta del servicio
- Inversión en Investigación y Desarrollo, lo que se refleja en mejor tecnología
- Economía de escala: los costos son compartidos entre todos los subscriptores

Sin el modelo de proveedor/consumidor, no todos los consumidores pueden invertir en mejora y alta tecnología, puesto que no es su *core business*,

¿QUÉ ES CLOUD COMPUTING?

A model for enabling ubiquitous, convenient, ondemand network access to a shared pool of configurable computing resources (networks, servers, storage, applications and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction)”

[National Institute of Standards and Technology (NIST)]

“Cloud computing ofrece el uso de recursos computacionales como un servicio [...] Es simplemente una gran infraestructura de computación distribuida que los usuarios pueden acceder por la red”

[Microsoft]

CARACTERÍSTICAS DE CLOUD COMPUTING

De acuerdo al NIST, cloud computing tiene las siguientes cinco características:

ON-DEMAND SELF-SERVICES

Los usuarios son capaces de proveer, monitorear y administrar los recursos que necesiten sin intervención del proveedor

BROAD NETWORK ACCESS

Los servicios computacionales se provee a través de la red y para dispositivos heterogéneos

MEASURED SERVICE

El uso de recursos se lleva a nivel de aplicación y ocupante. Provee al usuario y al proveedor de servicio, una forma de control sobre el uso de los recursos

RESOURCE POOLING

Los recursos de TI (networks, servers, storage, applications, servicios) son compartidos entre múltiples aplicación y ocupantes. Múltiples clientes comparten el mismo recurso físico

ELASTICITY

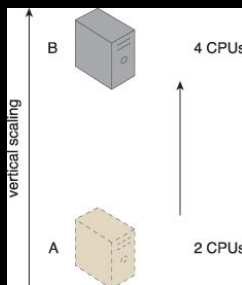
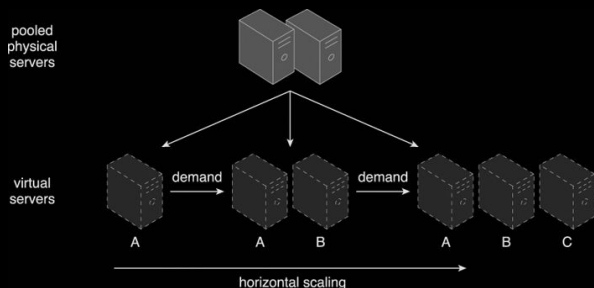
Los servicios deben ser capaces de escalar rápidamente según las necesidades de los usuarios



ESCALABILIDAD vs ELASTICIDAD

ESCALABILIDAD

Habilidad de los recursos de TI para manejar el incremento o decremento de la demanda de uso. Puede ser horizontal (agregar o quitar elementos del mismo tipo. *Scale in, scale out*) o puede ser vertical (reemplazar un elemento por otro con mayor o menor capacidad. *Scale up, scale down*)



ELASTICIDAD

Habilidad automatizada de la nube para transparentemente escalar recursos de TI en respuesta a condiciones de ejecución o a reglas predefinidas por el cliente.

Scalability is the option to in(de)crease resources where needed. Elasticity is the grade of ability to dynamically scale (possibly without interruptions of service).

BENEFICIOS PRINCIPALES DE CLOUD COMPUTING



Cost

Cloud computing eliminates the capital expense of buying hardware and software and setting up and running on-site datacenters—the racks of servers, the round-the-clock electricity for power and cooling, the IT experts for managing the infrastructure. It adds up fast.



Speed

Most cloud computing services are provided self service and on demand, so even vast amounts of computing resources can be provisioned in minutes, typically with just a few mouse clicks, giving businesses a lot of flexibility and taking the pressure off capacity planning.



Global scale

The benefits of cloud computing services include the ability to scale elastically. In cloud speak, that means delivering the right amount of IT resources—for example, more or less computing power, storage, bandwidth—right when it is needed and from the right geographic location.



Productivity

On-site datacenters typically require a lot of “racking and stacking”—hardware setup, software patching, and other time-consuming IT management chores. Cloud computing removes the need for many of these tasks, so IT teams can spend time on achieving more important business goals.



Performance

The biggest cloud computing services run on a worldwide network of secure datacenters, which are regularly upgraded to the latest generation of fast and efficient computing hardware. This offers several benefits over a single corporate datacenter, including reduced network latency for applications and greater economies of scale.



Reliability

Cloud computing makes data backup, disaster recovery and business continuity easier and less expensive because data can be mirrored at multiple redundant sites on the cloud provider's network.



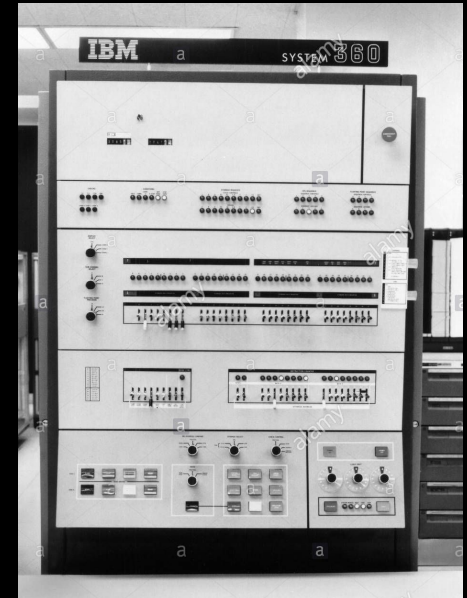
Security

Many cloud providers offer a broad set of policies, technologies and controls that strengthen your security posture overall, helping protect your data, apps and infrastructure from potential threats.

LA EVOLUCIÓN DE CLOUD COMPUTING

1960s


MAINFRAME ERA: Multiple users could share and connect to mainframes over basic serial connections using terminals. The mainframe was responsible for all the logic, storage, and processing of data, and the terminals connected to them had limited computational power, if any. These systems continued in widespread use for more than 30 years and, to some degree, continue to exist today



LA EVOLUCIÓN DE CLOUD COMPUTING

1960s

1990s




The global information age emerged, with the internet rapidly being adopted. Network bandwidth improved by many orders of magnitude, from ordinary dial-up access to dedicated fiber connectivity today. In addition, cheaper and more powerful hardware emerged. Furthermore, the evolution of the World Wide Web and dynamic websites necessitated multi-tier architectures.

Multitier architectures enabled the modularization of software by separating the application presentation, application logic, and storage as individual entities. With this modularization and decoupling, it was not long before these individual software entities were running on distinct physical servers (typically due to differences in hardware and software requirements). This led to an increase of individual servers in organizations; however, it also led to poor average utilization of server hardware

LA EVOLUCIÓN DE CLOUD COMPUTING


1960s

1990s



The global information age emerged, with the internet rapidly being adopted. Network bandwidth improved by many orders of magnitude, from ordinary dial-up access to dedicated fiber connectivity today. In addition, cheaper and more powerful hardware emerged. Furthermore, the evolution of the World Wide Web and dynamic websites necessitated **multi-tier** architectures.

Tier vs Layer?




Multitier architectures enabled the modularization of software by separating the application presentation, application logic, and storage as individual entities. With this modularization and decoupling, it was not long before these individual software entities were running on distinct physical servers (typically due to differences in hardware and software requirements). This led to an increase of individual servers in organizations; however, it also led to poor average utilization of server hardware

LA EVOLUCIÓN DE CLOUD COMPUTING

1960s

1990s

2000s



Virtual machine technology matured well enough in the 2000s to become available as commercial software. Virtualization enables an entire server to be encapsulated as an image, which can be run seamlessly on hardware and enable multiple virtual servers to run simultaneously and share hardware resources. Virtualization thus enables servers to be consolidated, which accordingly improves system utilization.

Simultaneously, grid computing gained traction in the scientific community in an effort to solve large-scale problems in a distributed fashion. With grid computing, computer resources from multiple administrative domains work in unison for a common goal. Grid computing brought forth many resource-management tools (for example, schedulers and load balancers) to manage large-scale computing resources.

As the various computing technologies evolved, so did the economics of computing. Even during the early days of mainframe-based computing, companies such as IBM offered to host and run computers and software for various organizations, such as banks and airlines. In the internet age, third-party web hosting also became popular. With virtualization, however, providers have unparalleled flexibility in accommodating multiple clients on a single server, sharing hardware and resources between them.

MODELOS DE DESPLIEGUE

Hay tres modelos de despliegue o tipos de clouds: pública, privada e híbrida

- **Pública:** propiedad de un proveedor de cloud pero disponible al público
- **Privada:** es propiedad de una organización o ente específico, quien controla el acceso a la misma. Puede o no ser on-premise.
- **Híbrida:** es la combinación de cloud pública y privada

¿QUÉ ES MULTI CLOUD?

No es un modelo de despliegue. Es el uso de dos o más clouds de proveedores diferentes. Previene el vendor lock-in. ¿Qué problemas puede traer?

MODELOS DE DESPLIEGUE: PUBLIC CLOUD

La infraestructura física es propiedad de un proveedor (Amazon, Microsoft, Google) y es accesible al público a través de internet. Los usuarios se conectan remotamente sin necesidad de comprar y configurar el ambiente.

- Los recursos computacionales típicamente se comparten (multi-tenant), aunque también se pueden adquirir servidores físicos (*bare metal*) dedicados
- A los usuarios se les cobra por el tiempo que usen dichos servicios
- Los principales proveedores ofrecen muchos servicios gratuitamente

El proveedor es responsable de los data centers, hardware y cualquier otro elemento de la infraestructura

MODELOS DE DESPLIEGUE: PRIVATE CLOUD

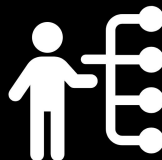
Es un ambiente de computación en la que todo el hardware y software son dedicados exclusivamente y accesibles únicamente a un solo cliente/organización.

Hay dos tipos de private cloud: on-premise y managed private clouds



ON-PREMISE

Los recursos de Software y Hardware están físicamente en el centro de datos de la organización. Dicha organización es totalmente responsable de mantener la infraestructura



MANAGED CLOUD

Alojada en un centro de datos propiedad de un proveedor externo. El modelo de administración puede ser en diferentes niveles desde totalmente administrada por el proveedor o combinado

¿Por qué escoger private cloud?

Para muchas organizaciones es la forma más fácil (o la única) para cumplir con requerimientos legales. Otros la escogen para proteger documentos confidenciales, propiedad intelectual, registros médicos.

MODELOS DE DESPLIEGUE: PRIVATE CLOUD

Para que un private cloud sea realmente un “cloud”, debe tener ciertas tecnologías o características. Algunas relevantes son:



VIRTUALIZACIÓN

Permite abstraer los recursos del hardware físico y agrupados en pools de computación, almacenamiento, memoria y redes, los cuales se pueden distribuir entre múltiples VMs, containers y otros elementos virtuales. Permite maximizar el uso de hardware y facilita la escalabilidad y elasticidad de la nube.



SOFTWARE ADMINISTRATIVO

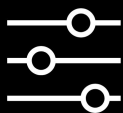
Provee un punto de control centralizado sobre la infraestructura y las aplicaciones que se ejecuten sobre estas. Esto permite optimizar la seguridad, disponibilidad y uso de recursos.



AUTOMATIZACIÓN

Permite acelerar tareas como aprovisionamiento de servidores, que de otra forma tendría que realizar manualmente y constantemente. La automatización reduce la dependencia humana y permite el autoservicio.

BENEFICIOS DE PRIVATE CLOUD



CONTROL COMPLETO SOBRE HARDWARE Y SOFTWARE

Libertad de comprar el hardware y software que la organización prefiera y no la impuesta por el proveedor de la nube. La organización puede personalizar el software a su gusto.



MAYOR CONTROL DE SEGURIDAD

Dado que todas las cargas computacionales se ejecutan detrás del Firewall de la organización, es más fácil controlar la seguridad de hardware y software



CUMPLIMIENTO TOTAL DE REGULACIONES

La organización no se ve obligado a cumplir con las regulaciones que el proveedor debe cumplir y puede tener más control sobre los mecanismos para las regulaciones aplicables

PUBLIC vs PRIVATE CLOUD

Public cloud sacrifices much of the control and security of private cloud, but provides significant benefits in exchange:



Greater elasticity and scalability: With public cloud, a customer can add capacity in response to unexpended surges in traffic, without purchasing and installing new hardware.



Lower cost of entry: Most customers can begin using public cloud services without adding physical compute resources of their own.



Faster access to the latest technologies: In many cases, economies of scale enable cloud providers to offer the latest hardware and software faster than customers could if they had to purchase and install them themselves.

VIRTUAL PRIVATE CLOUD (VPC)

Is a service from a public cloud provider that creates a private cloud-like environment on public cloud infrastructure. In a VPC, virtual network functions and security features give a customer the ability to define and control a logically isolated space in the public cloud, mimicking the private cloud's enhanced security within a multi-tenant environment.

```
provider "aws" {  
  region      = "${var.region}"  
  access_key  = "${var.access_key}"  
  secret_key  = "${var.secret_key}"  
}  
  
# VPC resources: This will create 1 VPC with 4 Subnets, 1 Internet Gateway, 4 Route Tables.  
  
resource "aws_vpc" "default" {  
  cidr_block      = var.cidr_block  
  enable_dns_support = true  
  enable_dns_hostnames = true  
}  
  
resource "aws_internet_gateway" "default" {  
  vpc_id = aws_vpc.default.id  
}  
  
resource "aws_route_table" "private" {  
  count = length(var.private_subnet_cidr_blocks)  
  
  vpc_id = aws_vpc.default.id  
}  
  
resource "aws_route" "private" {  
  count = length(var.private_subnet_cidr_blocks)  
  
  route_table_id      = aws_route_table.private[count.index].id  
  destination_cidr_block = "0.0.0.0/0"  
  nat_gateway_id      = aws_nat_gateway.default[count.index].id  
}
```



INFRASTRUCTURE AS CODE



HashiCorp

Terraform

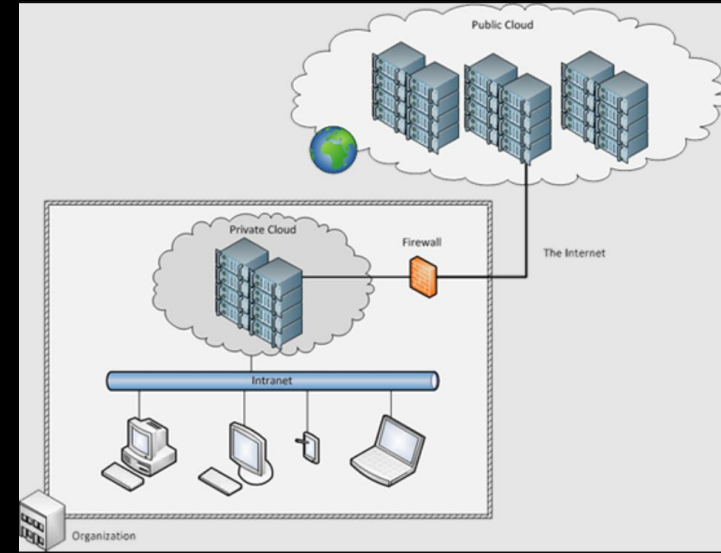


ANSIBLE

MODELOS DE DESPLIEGUE: HYBRID CLOUD

Es una infraestructura de TI que conecta la nube pública con una nube privada. Habilita a las organizaciones para:

- **Cumplir con regulaciones y seguridad:** datos sensibles/confidenciales se pueden reservar para la nube privada y mover a la pública los datos no sensibles
- **Adopción rápida de nuevas tecnologías:** la nube pública utiliza tecnología de punta que permite a la organización probar dichas tecnologías y si es factible, utilizarlas de inmediato para el desarrollo de nuevos productos
- **Optimización de recursos y ahorro de costos:** las cargas de trabajo variables pueden moverse a la nube y pagar solo por lo que se usa



MODELOS DE SERVICIO

Los tres modelos de servicio de cloud computing son: IaaS (Infrastructure as a Service), PaaS (Platform as a Service) y SaaS (Software as a Service). Cada tipo implica diferentes responsabilidades para el proveedor y para la organización

Cloud Computing Services: Who Manages What?

	Traditional IT	IaaS	PaaS	Serverless	SaaS
Applications	You manage	You manage	You manage	You manage	Provider manages
Data	You manage	You manage	You manage	Provider manages	Provider manages
Runtime	You manage	You manage	Provider manages	Provider manages	Provider manages
Middleware	You manage	You manage	Provider manages	Provider manages	Provider manages
OS	You manage	Provider manages	Provider manages	Provider manages	Provider manages
Virtualization	You manage	Provider manages	Provider manages	Provider manages	Provider manages
Servers	You manage	Provider manages	Provider manages	Provider manages	Provider manages
Storage	You manage	Provider manages	Provider manages	Provider manages	Provider manages
Networking	You manage	Provider manages	Provider manages	Provider manages	Provider manages

 You manage  Provider manages

SOFTWARE AS A SERVICE

Es software de aplicación alojado en la nube y que es accedido a través del browser, un cliente de escritorio dedicado o un API. Ejemplos clásicos: Gmail, Jira, Office 365, Google Drive, Netflix, Disney+, LucidChart, entre otros.

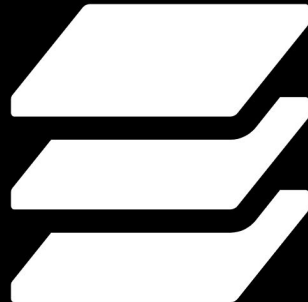
- Los usuarios pagan una suscripción mensual, anual o por uso
- Updates automáticos ocurren tan pronto como el proveedor los habilite sin necesidad de acción por parte del usuario
- Los datos de los usuarios son respaldados constantemente, por lo que pérdidas de datos son prácticamente imposibles

SaaS es el modelo tradicional para la mayoría del software comercial actual.

PLATFORM AS A SERVICE

Provee a los developers un stack completo (hardware, software, herramientas de desarrollo) para ejecutar, desarrollar y administrar aplicaciones sin el costo, complejidad e inflexibilidad de mantener la plataforma tradicionalmente

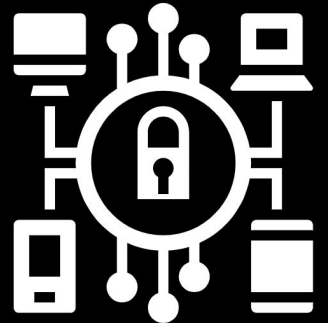
- El proveedor “hostea” todo, incluyendo middleware, SDKs, bases de datos, CI/CD
- En la actualidad PaaS se construye alrededor de containers
- Habilita a los developers para probar nuevas tecnologías de una forma rápida y sin verse obligados a aprender todos los detalles sobre instalación/configuración



INFRASTRUCTURE AS A SERVICE

Provee acceso por demanda a recursos computacionales esenciales (servidores físicos y virtuales, redes y almacenamiento) a través de internet bajo un modelo de pago por uso

- Nivel de control de más bajo nivel para los usuarios
- Fue el modelo más popular al inicio de Cloud Computing moderno



SERVERLESS COMPUTING

Ejecuta código de aplicación en una modalidad por request y escala la infraestructura según la cantidad de requests

- Los clientes sólo pagan por los recursos que utilicen por el tiempo en el que se ejecute el código.
- FaaS (Function as a service) es un subconjunto de Serverless. FaaS ejecuta código en respuesta a eventos específicos.

Serverless utiliza servidores y cualquier otra infraestructura necesaria

TECNOLOGÍAS HABILITADORES DE CLOUD COMPUTING



Broadband Networks and
Internet Architecture



Data Center Technology



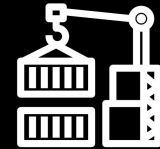
Web Technology



Virtualization Technology



Multitenant Technology



Containerization

TECNOLOGÍAS HABILITADORAS: VIRTUALIZACIÓN

Virtualization is the process of converting a physical IT resource into a virtual IT resource. Most types of IT resources can be virtualized, including:

- Servers
- Storage disks
- Network (VLANs)
- Power (virtual UPSs)

The first step in creating a new virtual server through virtualization software is the allocation of physical IT resources, followed by the installation of an operating system.

Virtual servers use their own guest operating systems, which are independent of the operating system in which they were created.

TECNOLOGÍAS HABILITADORAS: VIRTUALIZACIÓN

Both the guest operating system and the application software running on the virtual server are unaware of the virtualization process, meaning these virtualized IT resources are installed and executed as if they were running on a separate physical server

Virtualization software runs on a physical server called a *host* or *physical host*, whose underlying hardware is made accessible by the virtualization software.

This software is sometimes referred to as a virtual machine manager or a virtual machine monitor (VMM), but most commonly known as a *hypervisor*

Through hardware independence, virtual servers can easily be moved to another virtualization host, automatically resolving multiple hardware-software incompatibility issues.

As a result, cloning and manipulating virtual IT resources is much easier than duplicating physical hardware.

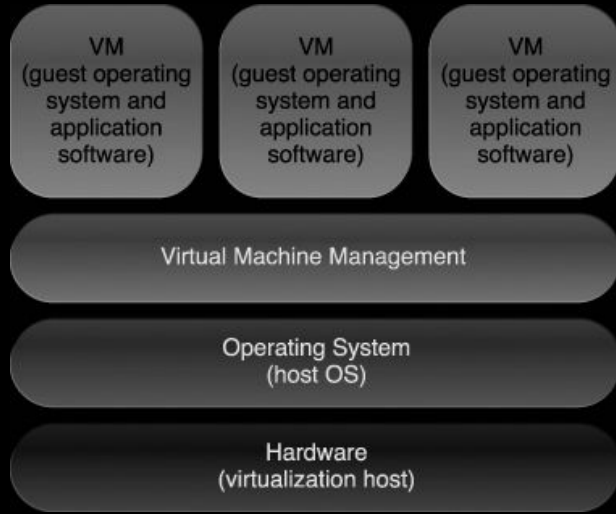
TECNOLOGÍAS HABILITADORAS: VIRTUALIZACIÓN

Virtual servers are created as virtual disk images that contain binary file copies of hard disk content. This ease of manipulation and replication is one of the most salient features of virtualization technology as it enables:

- The creation of standardized virtual machine images commonly configured to include virtual hardware capabilities, guest operating systems, and additional application software, for pre-packaging in virtual disk images in support of instantaneous deployment.
- Increased agility in the migration and deployment of a virtual machine's new instances by being able to rapidly scale out and up.
- The ability to roll back, which is the instantaneous creation of VM snapshots by saving the state of the virtual server's memory and hard disk image to a host-based file. (Operators can easily revert to these snapshots and restore the virtual machine to its prior state.)
- The support of business continuity with efficient backup and restoration procedures, as well as the creation of multiple instances of critical IT resources and applications.

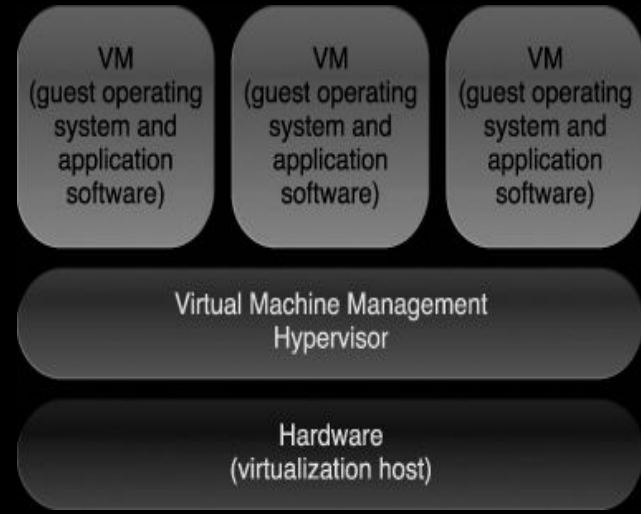
OPERATING SYSTEM BASED VIRTUALIZATION

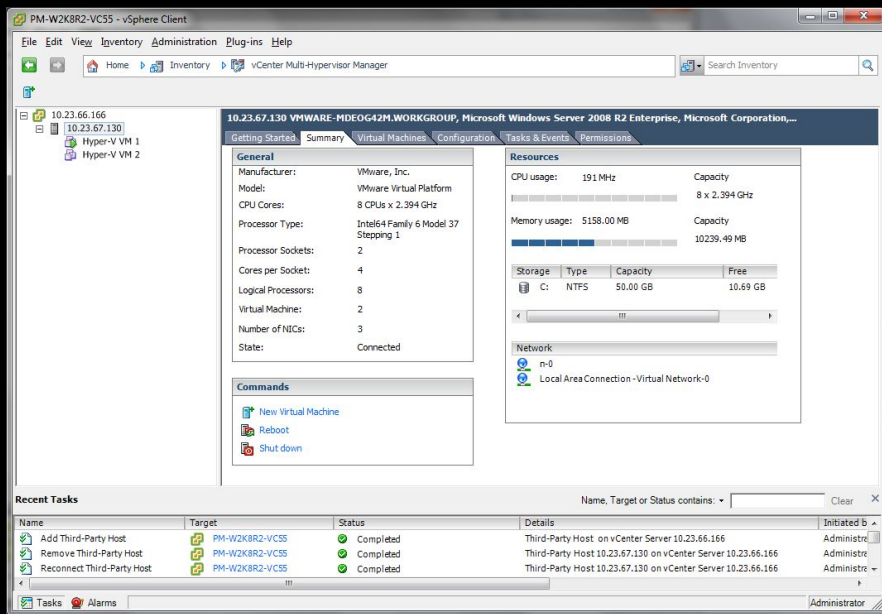
Operating system-based virtualization is the installation of virtualization software in a pre-existing operating system, which is called the *host operating system*



HARDWARE-BASED VIRTUALIZATION

Installation of virtualization software directly on the physical host hardware so as to bypass the host operating system, which is presumably engaged with operating system-based virtualization.





A hypervisor has a simple user-interface that requires a negligible amount of storage space. It exists as a thin layer of software that handles hardware management functions to establish a virtualization management layer.

Device drivers and system services are optimized for the provisioning of virtual servers, although many standard operating system functions are not implemented.

This type of virtualization system is essentially used to optimize performance overhead inherent to the coordination that enables multiple virtual servers to interact with the same hardware platform.

TECNOLOGÍAS HABILITADORAS: CONTENEDORES

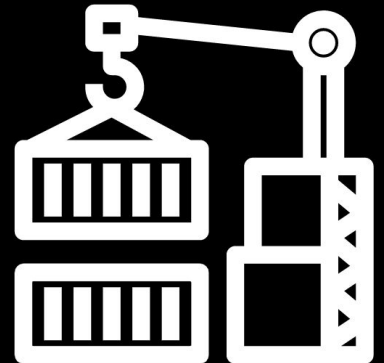
Containerization is an operating system-level virtualization technology used to deploy and run applications and cloud services without the need to deploy a virtual server for each solution

Using containers enables multiple isolated cloud services to run on a single physical server or virtual server while accessing the same operating system kernel.

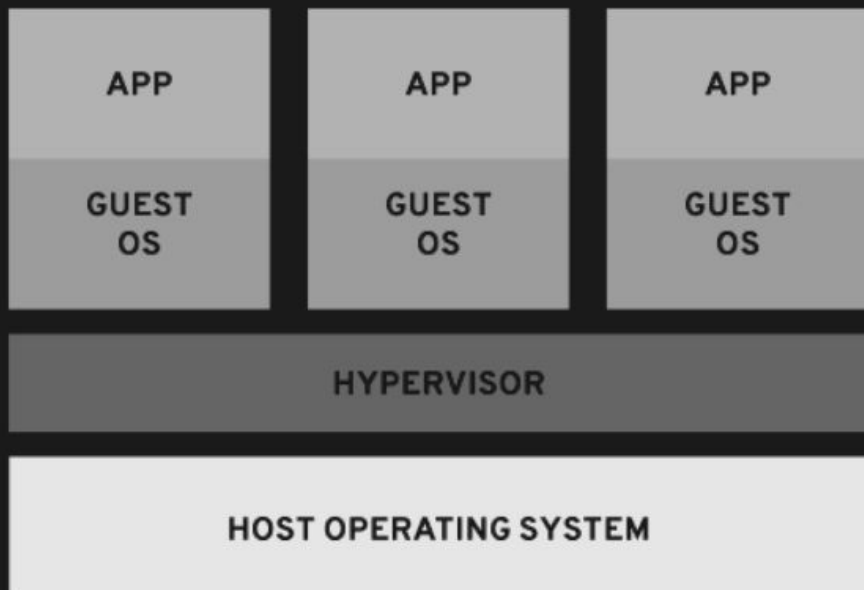
Containers are an abstraction at the application or service layer that package code and dependencies together

The operating system kernel allows for the existence of multiple isolated user-space instances or multiple isolated runtimes known as containers, partitions, virtual engines, jails or chroot jails.

An application running inside a container can only see the container's contents and devices attached to the container

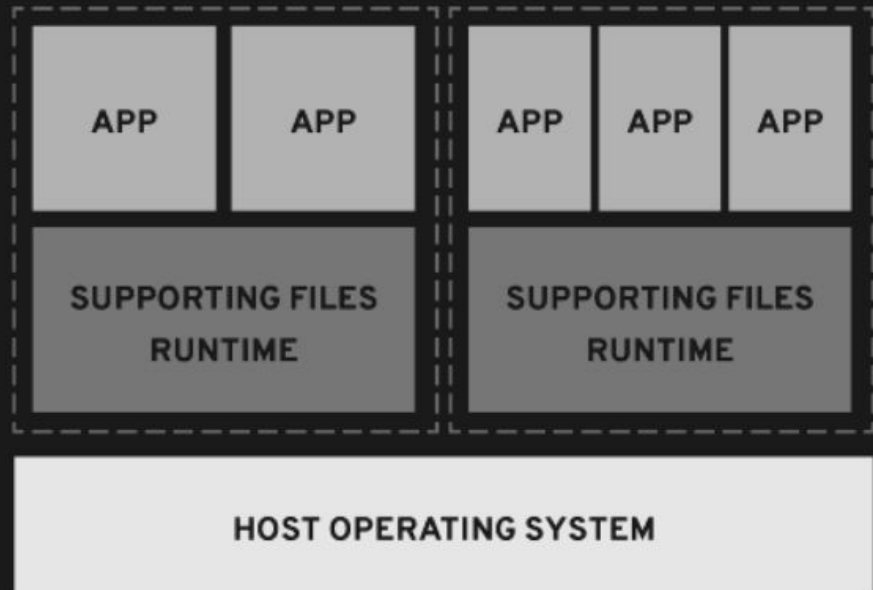


VIRTUALIZATION



VS.

CONTAINERS



BENEFICIOS DE CONTENEDORES

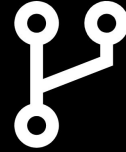
Portability is one of the key benefits of containers, allowing cloud resource administrators to move containers to any environment that shares the same host operating system and container engine that the container is hosted on, and without the need to change the application or software



Efficient resource utilization is achieved by significantly reducing the CPU, memory and storage usage footprint compared to virtual servers.

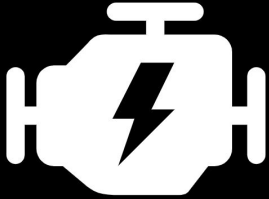


Containers can be created and deployed much faster than virtual servers, which supports a more agile process and facilitates continuous integration.



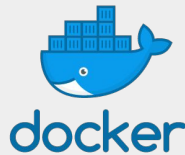
Containers allow versions of an software code and its dependencies to be tracked. Some container images provide the capability of a manifest file that allows cloud service owners and developers to maintain and track versions of a container and its software

ELEMENTOS FUNDAMENTALES DE CONTENEDORES



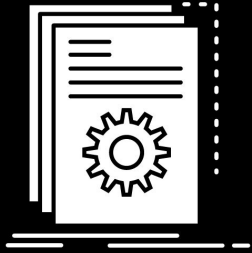
Container Engine

The key component of container architecture is the *container engine*, also referred to as the *containerization engine*. The container engine is specialized software that is deployed in an operating system to abstract the required resources and enable the definition and deployment of containers. Container engine software can be deployed on physical machines or virtual machines. Each container engine provides a set of management tools and commands/APIs to create, modify, schedule, run, stop, start or delete the containers.



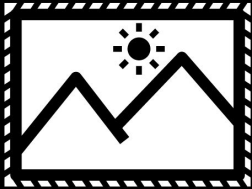
Apache
MESOS™

ELEMENTOS FUNDAMENTALES DE CONTENEDORES



Container Build File

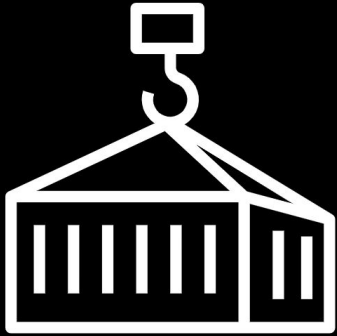
A *container build* file is a descriptor (created by the user or service) that represents the requirements of the application and services that run inside the container, as well as the configuration parameters required by the container engine in order to create and deploy the container. The syntax and format of the container build file and configuration parameters it defines depend on the choice of container engine.



Container Image

The container engine uses a *container image* to deploy an image based on pre-defined requirements. Based on the defined descriptions, the container engine customizes the operating system image and the required commands or services for the application. This customized image is normally an immutable read-only image, which enables the deployed application or services in the container to function and perform tasks, but prevents any changes from being made.

ELEMENTOS FUNDAMENTALES DE CONTENEDORES

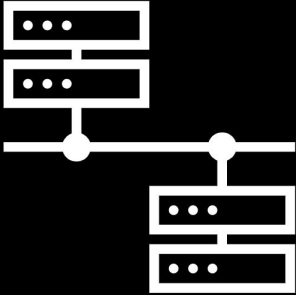


Container

The *container* is an executable instance of a pre-defined or customized container image that contains one or more software programs, most commonly an application or service. While containers are isolated from each other, they may be required to access a shared resource over the network, such as a file system or remote IT resource. This is possible without impacting the isolated containers.

Each container may have one application or process running in it. Containers can also host multiple applications, services or processes. Applications deployed in a container are typically scheduled with the container, meaning that they start and stop with the container.

ELEMENTOS FUNDAMENTALES DE CONTENEDORES



Networking Address

Each container has its own *network address* (such as an IP address) used to communicate with other containers and external components. A container can be connected to more than one network by allocating additional network addresses to the container.

Containers use the physical or virtual network card of the system that the container engine is deployed on to communicate with other containers and IT resources. When multiple applications need to be deployed and isolated, containers are used to isolate the applications from each other while still sharing an IP address, the containers can be deployed in a pod. Though the sharing of storage devices between containers within a pod is optional, all containers inside the pod share the same IP address.

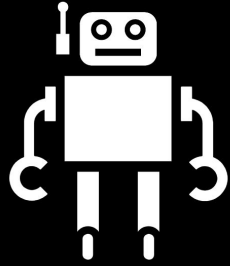
ELEMENTOS FUNDAMENTALES DE CONTENEDORES



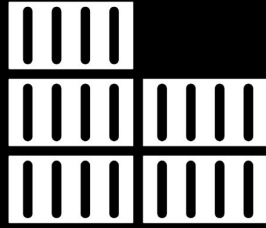
Storage Device

Similar to the networking address, a container may connect to one or more storage devices that are made available to the containers over the network. Each container has its own level of access to the storages defined by the system or administrators.

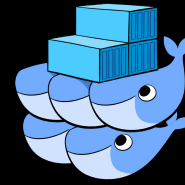
ORQUESTACIÓN DE CONTENEDORES



Automates the deployment, management, scaling, and networking of containers. Enterprises that need to deploy and manage hundreds or thousands containers



Container orchestration tools provide a framework for managing containers and microservices architecture at scale.



kubernetes

There are many container orchestration tools that can be used for container lifecycle management. Some popular options are Kubernetes, Docker Swarm, and Apache Mesos.

ORQUESTACIÓN DE CONTENEDORES: KUBERNETES (k8s)

Kubernetes is an open source container orchestration tool that was originally developed and designed by engineers at Google. Google donated the Kubernetes project to the newly formed Cloud Native Computing Foundation in 2015.

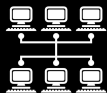
Kubernetes orchestration allows you to build application services that span multiple containers, schedule containers across a cluster, scale those containers, and manage their health over time

Kubernetes eliminates many of the manual processes involved in deploying and scaling containerized applications. You can cluster together groups of hosts, either physical or virtual machines, running Linux containers, and Kubernetes gives you the platform to easily and efficiently manage those clusters.

These clusters can span hosts across public, private, or hybrid clouds. For this reason, Kubernetes is an ideal platform for hosting cloud-native apps that require rapid scaling.

ORQUESTACIÓN DE CONTENEDORES: KUBERNETES (k8s)

The main components of k8s are:



Cluster: A control plane and one or more compute machines, or nodes.



Kubelet: This service runs on nodes and reads the container manifests and ensures the defined containers are started and running.



Control plane: The collection of processes that control Kubernetes nodes. This is where all task assignments originate.



Pod: A group of one or more containers deployed to a single node. All containers in a pod share an IP address, IPC, hostname, and other resources.

ORQUESTACIÓN DE CONTENEDORES: ¿CÓMO FUNCIONA?

When you use a container orchestration tool, such as Kubernetes, you will describe the configuration of an application using either a YAML or JSON file. The configuration file tells the configuration management tool where to find the container images, how to establish a network, and where to store logs.

When deploying a new container, the container management tool automatically schedules the deployment to a cluster and finds the right host, taking into account any defined requirements or restrictions. The orchestration tool then manages the container's lifecycle based on the specifications that were determined in the compose file.

You can use Kubernetes patterns to manage the configuration, lifecycle, and scale of container-based applications and services. These repeatable patterns are the tools needed by a Kubernetes developer to build complete systems.

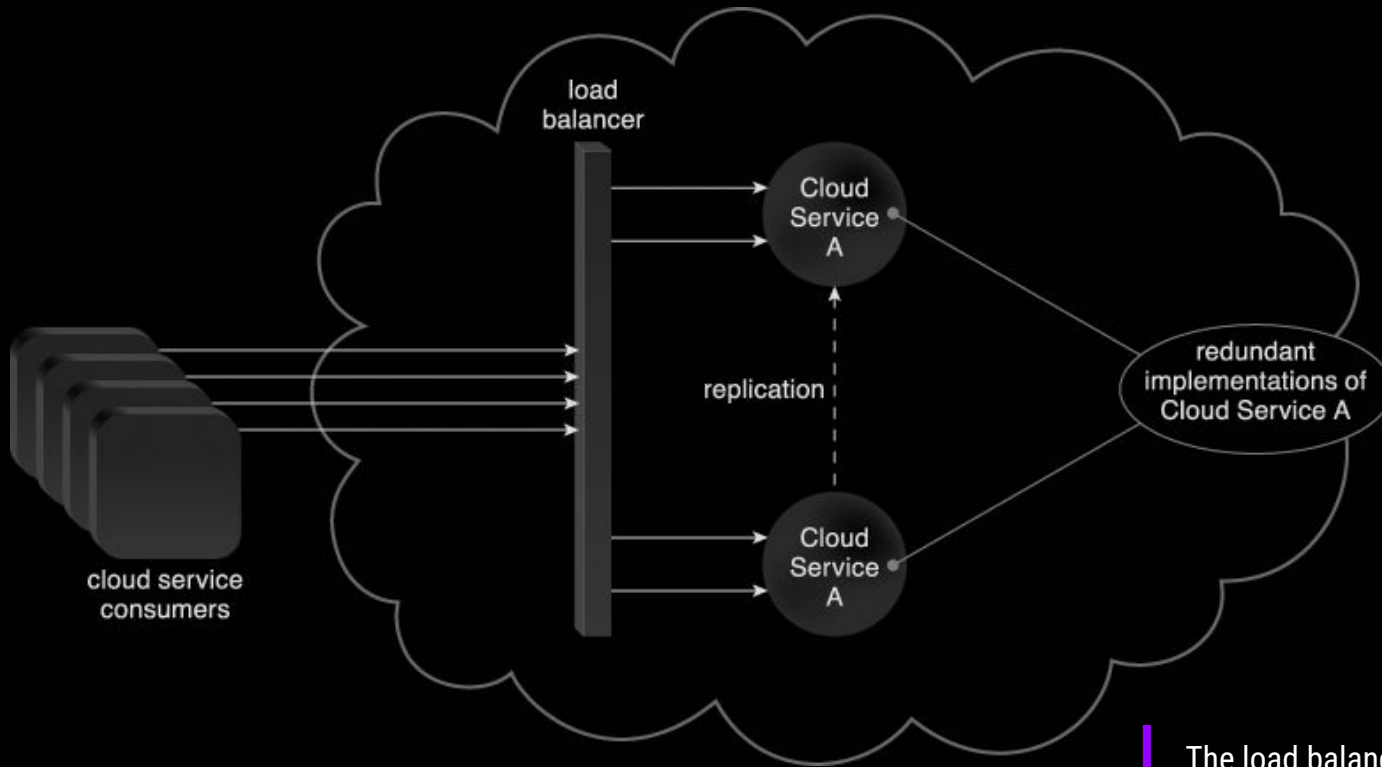
Container orchestration can be used in any environment that runs containers, including on-premise servers and public cloud or private cloud environments.

LOAD BALANCING

A common approach to horizontal scaling is to balance a workload across two or more IT resources to increase performance and capacity beyond what a single IT resource can provide. The *load balancer* mechanism is a runtime agent with logic fundamentally based on this premise.

Load balancers can perform a range of specialized runtime workload distribution functions that include:

- *Asymmetric Distribution* – larger workloads are issued to IT resources with higher processing capacities
- *Workload Prioritization* – workloads are scheduled, queued, discarded, and distributed workloads according to their priority levels
- *Content-Aware Distribution* – requests are distributed to different IT resources as dictated by the request content



The load balancer mechanisms can exist as a multi-layer network switch, dedicated hardware appliance, dedicated software-based system (common in server operating systems) or service agent (usually controlled by cloud management software)

CLOUD NATIVE

Cloud native refers less to where an application resides and more to how it is built and deployed.

- A cloud native application consists of discrete, reusable components known as microservices that are designed to integrate into any cloud environment.
- These microservices act as building blocks and are often packaged in containers.
- Microservices work together as a whole to comprise an application, yet each can be independently scaled, continuously improved, and quickly iterated through automation and orchestration processes.
- The flexibility of each microservice adds to the agility and continuous improvement of cloud-native applications.

CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape <https://github.com/cncf/landscape> has a growing number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator <https://www.cncf.io/training>

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified Service Provider <http://cncf.io/ksap>

C. Join CNCF's End User Community

For companies that don't offer cloud native services externally <http://cncf.io/enduser>

WHAT IS CLOUD NATIVE?

- **Operability:** Expose control of application/system lifecycle.
- **Observability:** Provide meaningful signals for observing state, health, and performance.
- **Elasticity:** Grow and shrink to fit in available resources and to meet fluctuating demand.
- **Resilience:** Fast automatic recovery from failures.
- **Agility:** Fast deployment, iteration, and reconfiguration.

www.cncf.io
info@cncf.io

v10

1. CONTAINERIZATION

- Normally done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices



3. ORCHESTRATION

- Pick an orchestration solution
- Kubernetes is the market leader and you should select a Certified Kubernetes Platform or Distribution
- <https://www.cncf.io/kk>



5. SERVICE MESH

- Connects services together and provides ingress from the Internet
- Service discovery, health checking, routing, load balancing
- Consider Envoy, Linkerd and CoreDNS



7. DISTRIBUTED DATABASE

When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding.



9. CONTAINER RUNTIME

You can use alternative container runtimes. The most common, all of which are OCI-compliant, are containerd, rkt and CRI-O.



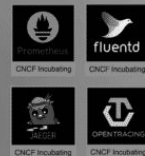
2. CI/CD

- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing



4. OBSERVABILITY & ANALYSIS

- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing compatible implementation like Jaeger



6. NETWORKING

To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net.



8. MESSAGING

When you need higher performance than JSON-RPC, consider using gRPC.



10. SOFTWARE DISTRIBUTION

If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.



The Twelve-Factor App

I. Codebase

One codebase tracked in revision control, many deploys

II. Dependencies

Explicitly declare and isolate dependencies

III. Config

Store config in the environment

IV. Backing services

Treat backing services as attached resources

V. Build, release, run

Strictly separate build and run stages

VI. Processes

Execute the app as one or more stateless processes

VII. Port binding

Export services via port binding

VIII. Concurrency

Scale out via the process model

IX. Disposability

Maximize robustness with fast startup and graceful shutdown

X. Dev/prod parity

Keep development, staging, and production as similar as possible

XI. Logs

Treat logs as event streams

XII. Admin processes

Run admin/management tasks as one-off processes

CLOUD COMPUTING

CE-5508

