

introducción

conceptos previos

ocultar/mostrar

- software libre
- GNU/Linux
 - comandos básicos
 - instalación software: apt, yum
 - shell scripts
 - ssh
- Seguridad y redes
 - DNS
 - direccionamiento IPv4
 - conceptos básicos redes
 - NAT
 - Bridges Linux
 - Redes virtuales
- Seguridad
 - usuarios, privilegios, sudo
 - 666 o 777
 - instalación software
 - concepto cortafuegos: iptables
- Editor de textos
 - en consola
 - VIM
 - emacs (emacs-nox)
 - atom, sublime-text
- Programación

qué es el cloud computing?

ocultar/mostrar

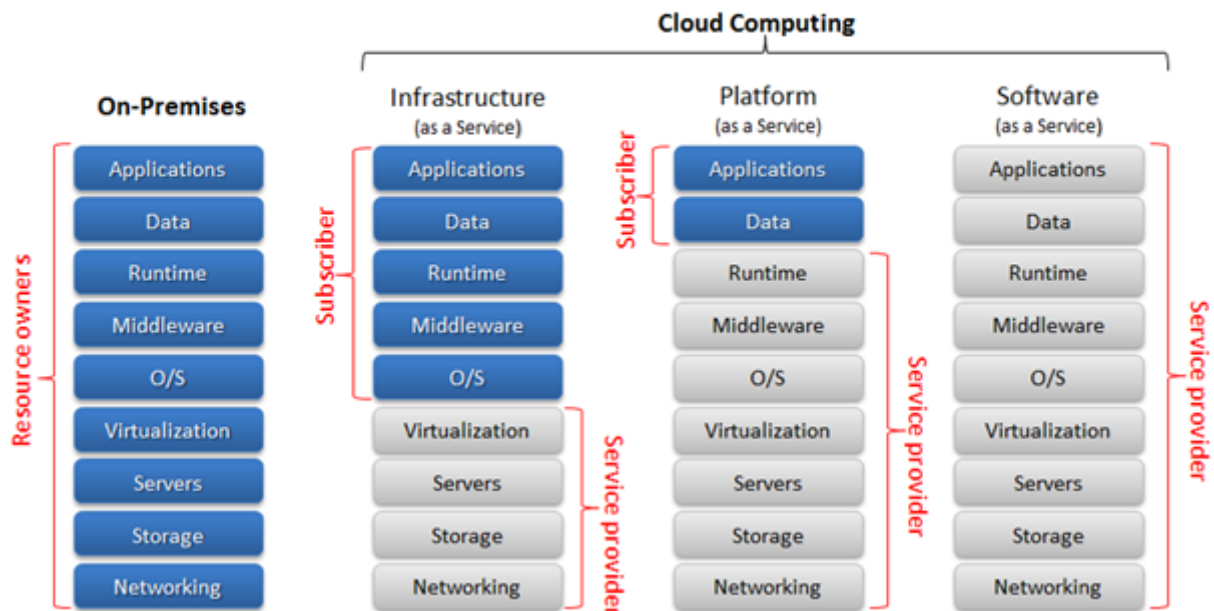
NIST

- organismo de EEUU, definición cloud computing
- servicio (ofrecer recursos) de forma automática y a demanda
- accesible a través de la red
 - público/privado
- modelo multi-tenancy (se comparten recursos con otros usuarios, pero se debe garantizar *aislamiento* y *seguridad*)
- los recursos/servicios se agrupan en pools
- elasticidad
 - usar recursos según a mis necesidades (subir y bajar)
 - escalabilidad: siempre sube
- Pago por uso

... as a Service

- modelo de negocio no basado en la venta de licencias o hardware
- oferta de servicios con características de la nube
- tradicionalmente se definen 3 capas
 - SaaS : Software as a Service
 - la capa que se ofrece al público
 - uso app a una web en lugar de tenerla instalada en el equipo
 - no todas las apps webs son SaaS, deben cumplir ciertas características
 - aplicaciones móviles que son front-end de aplicaciones SaaS
 - ejemplos: servicios Google, Office365, Dropbox
 - PaaS : Plataform as a Service
 - desarrollo web en la nube
 - utilizado por desarrolladores de soft
 - desarrollo + despliegue
 - ejemplos: Heroku, Google App Engine, Windows Azure, Openshift, CloudFoundry
 - capa fina de separación entre PaaS y IaaS
 - <https://www.paasfinder.org/compare>: comparador de plataformas PaaS
 - los contenedores ha dado en la línea de flotación de PaaS
 - IaaS : Infraestructure as a Service
 - Utilizado principalmente por administradores de sistemas
 - capacidad de cómputo, redes y diversos modos de almacenamiento
 - ejemplos: AWS, GCE, Azure, OpenStack

Separation of Responsibilities



Evolución de las aplicaciones

ocultar/mostrar

aplicación monolítica

- todos los componentes en el mismo nodo

- escalado vertical
 - aumentar recursos de la máquina en función de las necesidades (RAM, Disco, etc..)
 - limitaciones hardware
- arquitectura muy sencilla
 - los componentes se conectan entre ellos con recursos locales
- consideraciones de seguridad
 - el compromiso de un componente compromete a todos
- interferencias entre componentes
 - un componente funciona mal/fallar, afecta al resto de componentes
- complejidad de las actualizaciones
 - parada completa de la aplicación, aunque se tenga que actualizar solo un pequeño componente
- infraestructura estática y fija por años
- aplicación no tolerante a fallos
 - asume que la tolerancia a fallos la gestiona la capa inferior a la aplicación

aplicación distribuida

- idealmente un componente por nodo
 - a veces 2-3 componentes por nodo
 - a veces, un componente en varios nodos
- escalado horizontal
 - permite ampliar recursos sobre un componente en concreto
 - introduce la elasticidad (permite crecer y decrecer con facilidad)
- arquitectura más compleja
 - el desarrollador ha de tener en cuenta que están en nodos diferentes, mucho más complejo
- consideraciones de seguridad
 - es más complicado que un problema se extienda
- menos interferencias entre componentes
- simplicidad en las actualizaciones
 - más sencillo

¿SOA, cloud native o microservicios?

SOA

- Arquitectura orientada a servicio
 - servicios independientes
 - limitado a cierto tipo de aplicaciones (gran aplicación corporativa) → visión general que se tiene
 - orientado a desarrollo
- servicios independientes
- múltiples tecnologías interaccionando
- comunicación via WSDL y SOAP
- colas de mensajes
- se relaciona con aplicaciones corporativas

cloud native

- énfasis en la adaptación de la infraestructura a la demanda
 - orientado a que desde «sistemas» que se puede ofrecer a la aplicación
- uso extensivo de la elasticidad: infraestructura dinámica
- aplicaciones resilientes
 - resiliencia = capacidad de alguien/algo de responder ante la adversidad
 - la app se comunica con el nivel inferior para solicitar lo que necesite (Disco, RAM)
- elasticidad horizontal
- automatización

microservicios

- deriva del esquema SOA
- no existe definición formal
- servicios llevados a la mínima expresión (un proceso - un nodo): microservicios
- comunicación vía HTTP REST (API RESTFUL)
 - más fáciles de programar
 - difícil conversión de aplicaciones monolíticas a microservicios → nuevos desarrollos
- relacionado con procesos ágiles de desarrollo: entrega continua
- suele implementarse sobre contenedores
- <https://www.nginx.com/blog/introduction-to-microservices/>



From:

<https://matebcn.eu/> - **miguel angel torres egea**

Permanent link:

<https://matebcn.eu/info:cursos:openwebinars:intro-cloud-computing:introduccion>

Last update: **14/09/2018 11:35**

