Computación en la nube

Cloud computing

Cloud computing

- La computación en la nube (cloud computing) es un modelo que permite acceder a recursos informáticos (como servidores, almacenamiento, bases de datos, redes, software) a través de Internet
  - Consiste en usar la tecnología **como un servicio**, en lugar de poseerla y mantenerla directamente

Cloud computing

- Estos recursos son proporcionados por un proveedor de servicios en la nube, que los ofrece de forma:
  - Flexible: se pueden ajustar según las necesidades del momento
    - Un día se utiliza una máquina virtual con 4 GB de RAM, y al día siguiente se puede cambiar por una con 8 GB de RAM; solo se ajusta la configuración
  - Escalable: pueden aumentar o reducirse automáticamente
    - Tenemos una aplicación web que recibe más visitas en el verano: el servicio cloud creará más instancias de la aplicación automáticamente; cuando baje el tráfico, reducirá esas instancias
  - Bajo demanda: se usan y se pagan solo cuando se necesitan
    - Usamos una instancia de cómputo solo durante las horas de trabajo, y la apagamos por la noche; la factura solo reflejará esas horas

Cloud computing

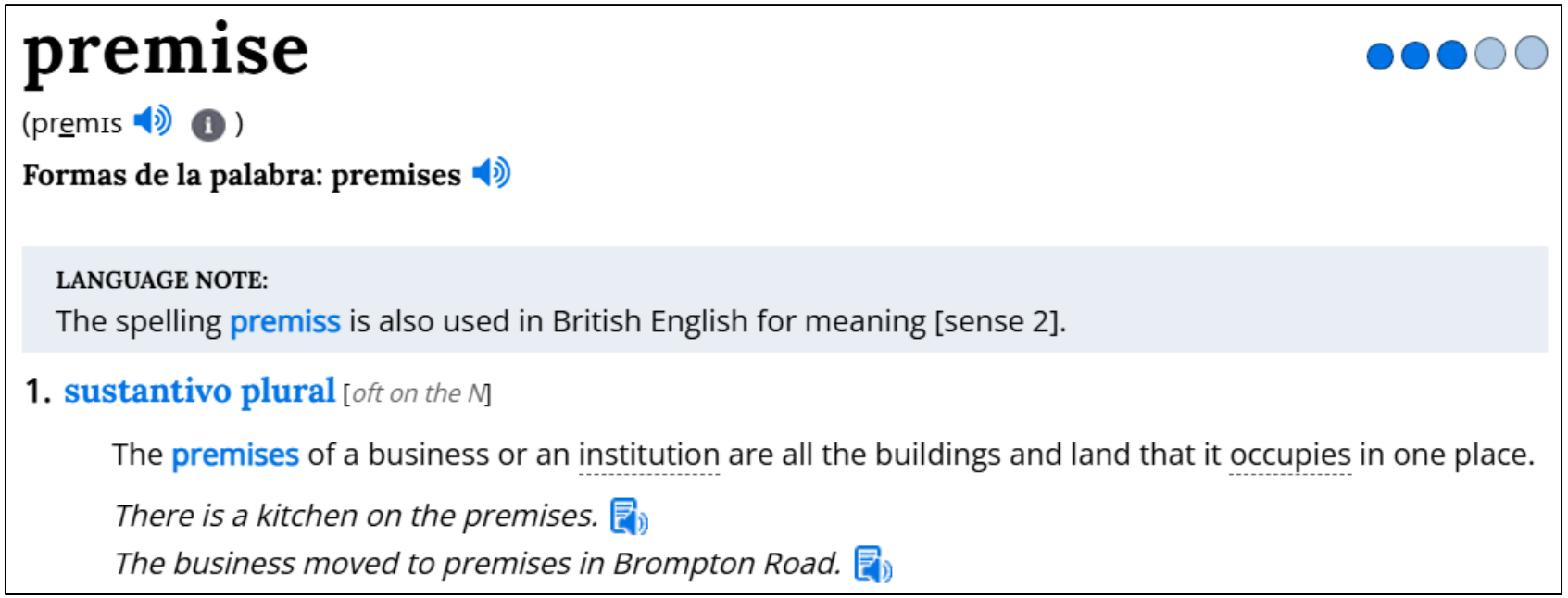
- Ventajas
  - Reduce los costes de la infraestructura
  - Aumenta la agilidad y el tiempo de salida al mercado de una empresa
  - Permite escalar los recursos según se necesiten (bajo demanda)
  - Seguridad (relativa) del hardware y de los datos
    - Aunque no hay, por defecto, una copia de seguridad de los datos
- Inconvenientes
  - Dependencia del proveedor
    - Una empresa completamente desplegada en la nube está en manos del proveedor y sus condiciones

Cloud computing

- El cloud computing cambia el modelo de negocio
  - En vez de gestionar los recursos uno mismo, los gestiona el proveedor de cloud
  - El proveedor nos los entrega como **servicios** flexibles, escalables y bajo demanda
- Ejemplo
  - Modelo de negocio tradicional: si nuestra empresa compra un servidor y lo instalamos en la oficina, estaremos usando un **centro de datos local**
    - Esto se conoce como **on-premises** (ver diapositiva siguiente)
  - Modelo de negocio cloud: si nuestra empresa utiliza AWS para tener un servidor, ese servidor estará presente en un **centro de datos de Amazon**, que nosotros percibiremos como “la nube”

On-premises

- Este término se usa mucho en cloud, y significa, literalmente, “en las dependencias”, “en las instalaciones”
- *Collins dictionary* define `premise` de este modo:



premise

(prɛmɪs)

Formas de la palabra: premises

LANGUAGE NOTE:
The spelling premiss is also used in British English for meaning [sense 2].

1. **sustantivo plural** [*oft on the N*]

The premises of a business or an institution are all the buildings and land that it occupies in one place.

There is a kitchen on the premises.
The business moved to premises in Brompton Road.

Principales proveedores de cloud computing

- Amazon Web Services (AWS)
- Microsoft Azure
  - *Azure* es una métafora de la nube
    - Azure /ˈæʒər/: a deep blue, occasionally somewhat purple, similar to the color of a clear blue sky

https://www.collinsdictionary.com/es/diccionario/ingles/azure

- Google Cloud Platform (GCP)
- IBM Cloud
- Oracle Cloud Infrastructure (OCI)
- Alibaba Cloud
- Huawei Cloud

Cloud computing

- El cloud computing se apoya en los centros de datos
  - Cloud computing no existe sin los centros de datos
    - Los servicios en la nube se ejecutan sobre los servidores que están en esos centros
  - La característica del cloud computing es que, en lugar de que el usuario tenga que:
    - Alquilar o comprar un local
    - Comprar y mantener sus propios servidores

    es el **proveedor de cloud** quien gestiona esos recursos y los ofrece vía Internet

Elementos esenciales de la arquitectura cloud

1. Modelos de despliegue
   - Describe **dónde se alojan** los recursos y servicios en la nube y quién tiene el control sobre ellos
2. Modelos de servicios
   - Describen **qué componentes** de la infraestructura y del software **gestiona** el proveedor de la nube y qué parte **gestiona** el cliente
3. Automatización y gestión como código fuente
   - Describe **cómo se implementan**, **configuran** y **administran** los servicios de forma automatizada mediante código fuente

1. Modelos de despliegue

1. Modelos de despliegue

- Describe **dónde y cómo se aloja** la infraestructura y los servicios en la nube
- El NIST (National Institute of Standards and Technology) define cuatro modelos de despliegue en su documento SP 800-145
  - Nube Pública (Public Cloud)
  - Nube Privada (Private Cloud)
  - Nube Híbrida (Hybrid Cloud)
  - Multinube (Multi-Cloud)

Nube pública

- La infraestructura y los servicios son propiedad de un proveedor externo y se comparten entre múltiples clientes
  - Algunos proveedores: AWS, Azure, Google Cloud
- Ventajas
  - Escalabilidad rápida y prácticamente ilimitada
  - Coste inicial bajo (pago por uso)
  - No requiere mantenimiento de hardware propio
- Inconvenientes
  - Muy poco control sobre la infraestructura
  - La seguridad y el cumplimiento normativo dependen del proveedor
  - Se comparten recursos físicos con otros clientes
    - Aunque los clientes están aislados virtualmente

Nube privada

- La infraestructura es propiedad de y está gestionada por una sola organización
  - Puede estar **on-premises** o en un proveedor externo dedicado
  - Algunos proveedores: OpenStack, VMware vSphere, Azure Stack
- Ventajas
  - Más control sobre la seguridad y el cumplimiento normativo
  - Personalización completa de la infraestructura
  - Ideal para datos sensibles o regulados
- Inconvenientes
  - Coste alto (hardware, mantenimiento, personal especializado)
  - Escalabilidad limitada a la inversión que se quiera o pueda hacer
  - Implementación y actualización más compleja

Nube híbrida

- Combina nube pública y privada, consiguiendo que ciertas cargas estén en la nube pública y otras en la privada
  - Ejemplo: Una empresa que mantiene datos sensibles en su nube privada, y usa AWS para servidores web
- Ventajas
  - Flexibilidad: permite mover cargas entre nubes, según necesidad
  - Optimiza costes y rendimiento
- Inconvenientes
  - Gestión más compleja (redes, seguridad, integración)
  - Posible latencia al mover datos entre nubes
  - Mayor necesidad de automatización y herramientas de orquestación

Multinube

- Uso de varios proveedores de nube pública para distribuir cargas y servicios
  - Ejemplo: Una empresa que usa AWS para servidores web, Google Cloud para IA/ML, y Azure para bases de datos corporativas
- Ventajas
  - Evita la dependencia de un solo proveedor
  - Optimiza costes y servicios, al poder elegir lo mejor de cada nube
  - Redundancia y resiliencia ante fallos de un proveedor
- Inconvenientes
  - Gestión más complicada (consolas, APIs y facturación distintas)
  - La Integración entre nubes puede ser costosa y lenta
  - Requiere personal altamente especializado

Resumen

Modelo	Control	Escalabilidad	Coste	Complejidad	Seguridad
Pública	Bajo	Alta	Bajo	Baja	Medio
Privada	Alto	Media	Alto	Alta	Alto
Híbrida	Medio	Alta	Medio	Media/Alta	Alto
Multinube	Medio	Muy alta	Medio/Alto	Muy alta	Medio/Alto

2. Modelos de servicios

IaaS, PaaS, SaaS

2. Modelos de servicio

- Los modelos de servicio en la nube describen **qué parte** de la infraestructura (o del software) **gestiona** el proveedor cloud y qué parte **gestionamos** nosotros
- Tres modelos principales
  - IaaS (Infrastructure as a Service)
  - PaaS (Platform as a Service)
  - SaaS (Software as a Service)

IaaS (Infrastructure as a Service)

- El proveedor nos da una infraestructura (hardware) básica virtualizada
  - Servidores: características de CPU; memoria RAM
  - Almacenamiento: discos virtuales; porciones de discos virtuales
  - Redes: subredes; balanceadores de carga; firewalls
  - Imágenes base de sistemas operativos: Windows Server; Red Hat
- Nosotros gestionamos todo lo demás (el software)
  - La instalación de software, de bases de datos, de aplicaciones, las configuraciones, etc.

IaaS (Infrastructure as a Service)

- Cómo se gestionan los sistemas operativos
  - El sistema operativo se incluye como parte del entorno virtualizado, **aunque no sea hardware**
  - El proveedor ofrece imágenes preconfiguradas de distintos sistemas operativos, y se encarga de mantener la infraestructura física
  - El proveedor nos permite elegir un sistema operativo, y nosotros nos encargamos de administrarlo y configurarlo dentro de la máquina virtual (instalamos parches, configuramos servicios, etc.)

IaaS (Infrastructure as a Service)

- IaaS es apropiado cuando
  - Queremos el control total del sistema operativo y del software
  - Tenemos equipos DevOps que configuran entornos a medida
- Ejemplos
  - Amazon EC2 (AWS)
  - Google Compute Engine
  - Microsoft Azure Virtual Machines
  - DigitalOcean Droplets

PaaS (Platform as a Service)

- El proveedor nos da una plataforma lista para desplegar aplicaciones, y nos despreocupamos de los servidores, del sistema operativo y de la infraestructura
  - El proveedor gestiona: el sistema operativo, los servidores, el posible escalado, etc.
- Nosotros gestionamos nuestro código fuente y las configuraciones de nuestro software
  - Creamos un sitio web y solo nos preocupamos de su creación, despliegue o escalado
  - Para crear aplicaciones en .NET, por ejemplo

PaaS (Platform as a Service)

- Es apropiado cuando
  - Queremos centrarnos en desarrollar y desplegar código, no en configurar servidores
  - Como desarrolladores, queremos productividad y automatización
- Ejemplos
  - Google App Engine: crear, desplegar y escalar aplicaciones web
  - Heroku: desarrollar y alojar apps web
  - AWS Elastic Beanstalk: automatizar la implementación y el escalado de aplicaciones
  - Azure App Service: crear y alojar aplicaciones web, API’s y backends móviles

SaaS (Software as a Service)

- El proveedor nos da una aplicación completa y lista para usar desde el navegador o una app
  - El proveedor gestiona la infraestructura, las administraciones, las instalaciones de software, etc.
- Nosotros nos limitamos a usar el software y a realizar posibles configuraciones de usuario
  - Es decir, no instalamos nada; solo usamos (y configuramos) el software

SaaS (Software as a Service)

- Es apropiado cuando
  - Solo necesitamos usar una aplicación, sin preocuparnos por servidores, instalaciones, configuraciones de red, o mantenimientos
- Ejemplos
  - Gmail: correo electrónico
  - Google Docs y Google Sheets: procesador de textos y hoja de cálculo
  - Google Drive: almacenamiento en la nube
  - Microsoft 365: Word, Excel, Outlook, Teams, etc., en la nube
  - Salesforce: CRM para gestión de clientes y ventas
    - Customer Relationship Management: Gestión de Relaciones con los Clientes
  - Zoom: videoconferencia
  - Dropbox: almacenamiento y compartición de archivos

Resumen

Característica	IaaS	PaaS	SaaS
Nivel de control	Alto	Medio	Bajo
Quién gestiona infraestructura	Proveedor	Proveedor	Proveedor
Quién gestiona la app	Nosotros	Nosotros parcialmente	Proveedor
Enfoque principal	Infraestructura	Desarrollo	Usuario final
Ejemplo típico	AWS EC2	Google App Engine	Gmail

3. Automatización y gestión como código fuente

3. Automatización y gestión como código fuente

- Para crear una infraestructura, el modelo habitual se aleja mucho del contacto personal con un proveedor
  - **Fuera de cloud**, lo normal es llamar por teléfono o visitar a un proveedor y pedirle servicios y recursos
  - **En cloud**, lo normal es hacerlo todo uno mismo, por Internet, a través de distintos métodos:
    1. Usar la interfaz web del proveedor y crear todo a base de clics: limitado
    2. Usar Command Line Interface (CLI) para crear recursos a base de comandos: difícil de mantener si las estructuras son grandes
    3. Usar **Infrastructure as Code** (IaC) para definir toda la infraestructura con código fuente: potente, requiere aprendizaje y claridad del código
  - Ejemplo
    - En AWS, podemos abrir la consola (método 1), crear una VPC (nube virtual privada), una base de datos y una instancia EC2 (Elastic Compute Cloud) sin escribir una sola línea de código

3. Automatización y gestión como código fuente

- Se basa en tratar la infraestructura, o la configuración, como si fuera un programa
  - Entendiendo “programa” como **código** fuente
- Para configurar una infraestructura, se escribe código que describe, crea y gestiona recursos **automáticamente**
  - Para ello, se utiliza un **lenguaje declarativo**
    - En el que se dice qué se quiere conseguir, pero no se indica cómo conseguirlo
    - (No tiene nada que ver con un lenguaje imperativo (como Python))
- Distintas aproximaciones
  - IaC, CaC, PaC, DaC: **aC** significa “**as Code**”: **como código** (fuente)

IaC (Infrastructure as Code)

- Permite definir una infraestructura en la nube, a través de un archivo que la describe
  - El sufijo **“as Code”** se refiere a la automatización y definición mediante **código fuente** de un proceso que, tradicionalmente, se hacía a mano: abastecer un lugar de hardware concreto
  - Supongamos que queremos:
    - Una red de 30 hosts
    - Un servidor (con Windows Server)
    - 20 PC’s (con Windows 11)
  - Todo eso se definirá como código fuente
    - (Ver ejemplos en la diapositiva siguiente)

1. Una red con 30 direcciones para hosts
2. Un servidor (con Windows Server)
3. 20 PC’s (con Windows 11)

1

```
# Configuración del proveedor AWS y región
provider "aws" {
  region = "us-east-1"
}

# Crear red virtual (VPC)
resource "aws_vpc" "main_vpc" {
  cidr_block = "10.0.0.0/27"
}

# Crear subred dentro de la VPC
resource "aws_subnet" "main_subnet" {
  vpc_id            = aws_vpc.main_vpc.id
  cidr_block        = "10.0.0.0/27"
  availability_zone = "us-east-1a"
}
```

2

```
resource "aws_instance" "windows_server" {
  ami           = "ami-0c02fb55956c7d316" #
  instance_type = "t3.medium"
  subnet_id     = aws_subnet.main_subnet.id
  tags = {
    Name = "ServidorWS"
  }
}
```

3

```
resource "aws_instance" "windows_clients" {
  count         = 20
  ami           = "ami-0123456789abcdef0" #
  instance_type = "t3.small"
  subnet_id     = aws_subnet.main_subnet.id
  tags = {
    Name = "PC-${count.index + 1}"
  }
}
```

IaC (Infrastructure as Code)

- Este sistema permite
  - Versionar la infraestructura (tal y como se hace con el software)
    - Infraestructura 1.0, Infraestructura 1.1, Infraestructura 2.0, etc.
  - Reproducir entornos con exactitud
    - La infraestructura está definida en un archivo; basta con ejecutarlo en cualquier computadora
  - Escalar y destruir recursos de forma automática
    - Si una infraestructura no se necesita más, se destruye con un comando

IaC (Infrastructure as Code)

- El código se escribe en un lenguaje declarativo
  - Dicho código describe *qué queremos*: servidores, redes, bases de datos, reglas de firewall
    - No dice cómo crear los recursos; solo define el estado deseado de la infraestructura
- Después, una herramienta **ejecuta el código** y se encarga de:
  - Comparar el estado actual de la infraestructura con el estado deseado definido en el archivo
  - Generar un plan de cambios (qué hay que crear, actualizar o eliminar)
  - Ejecutar esos cambios en el proveedor cloud de manera automática
  - Esta ejecución se conoce como "**desplegar la infraestructura (deployment)**"

Lenguajes declarativos y herramientas de despliegue

Lenguaje / formato	Herramienta	Características
HCL (HashiCorp Configuration Language)	Terraform	Lenguaje propio de HashiCorp
JSON	Terraform, AWS CloudFormation	Terraform puede leer JSON; CloudFormation usa JSON o YAML
YAML	CloudFormation, Kubernetes, Ansible, GitHub Actions	Muy legible; no es un “lenguaje de programación” completo
Bicep	Azure Resource Manager (ARM)	Lenguaje declarativo de Microsoft

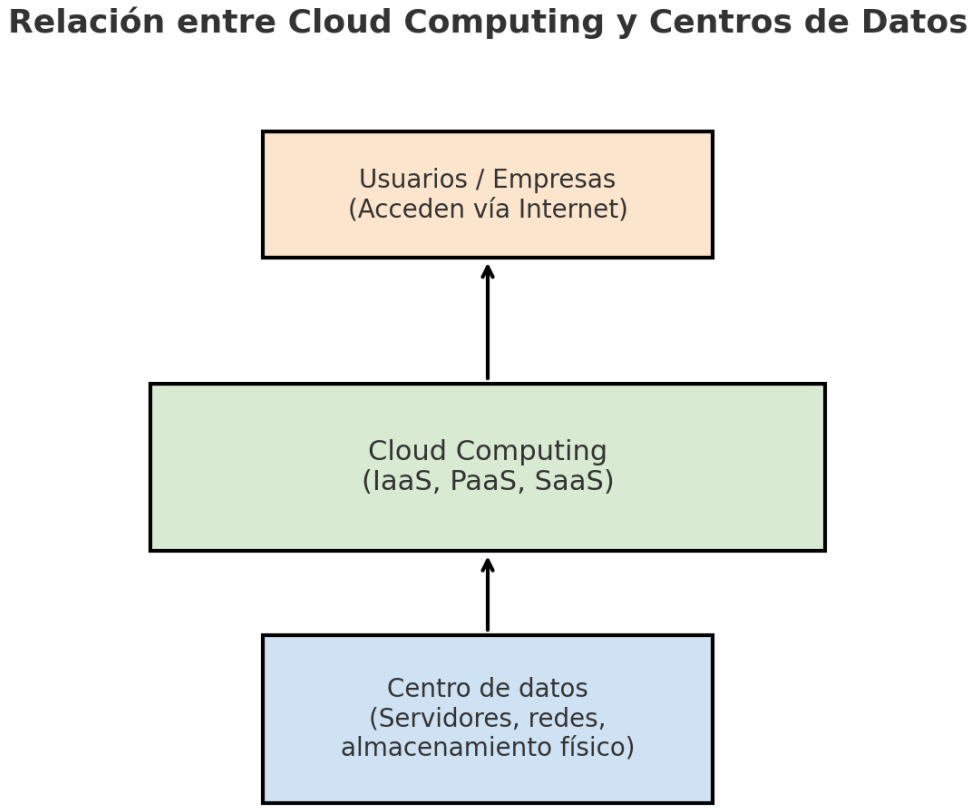
Otras aproximaciones

- CaC (Configuration as Code - Configuración como código)
  - Automatizar la configuración del sistema operativo o de las aplicaciones
- PaC (Policy as Code - Políticas como código)
  - Definir reglas de seguridad
- DaC (Data as Code - Datos como código)
  - Gestionar estructuras de datos

La nube sobre el mundo físico

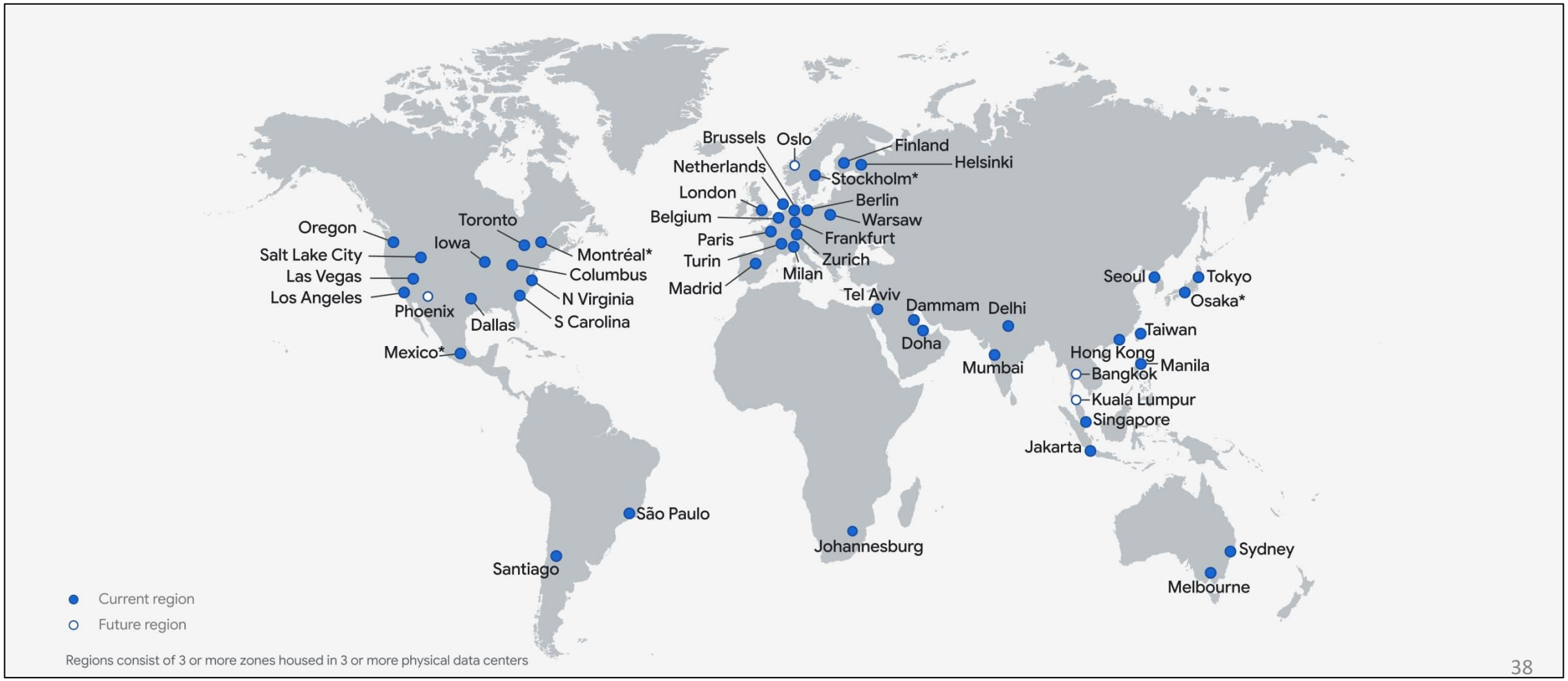
Cloud como capa de abstracción

- Podemos ver el cloud como la capa de servicios (IaaS, PaaS, SaaS) que abstrae al usuario de la complejidad de los centros de datos físicos
  - El usuario ve “instancias de VM, almacenamiento, bases de datos, aplicaciones en la nube”, pero realmente todo eso está corriendo en un centro de datos distribuido



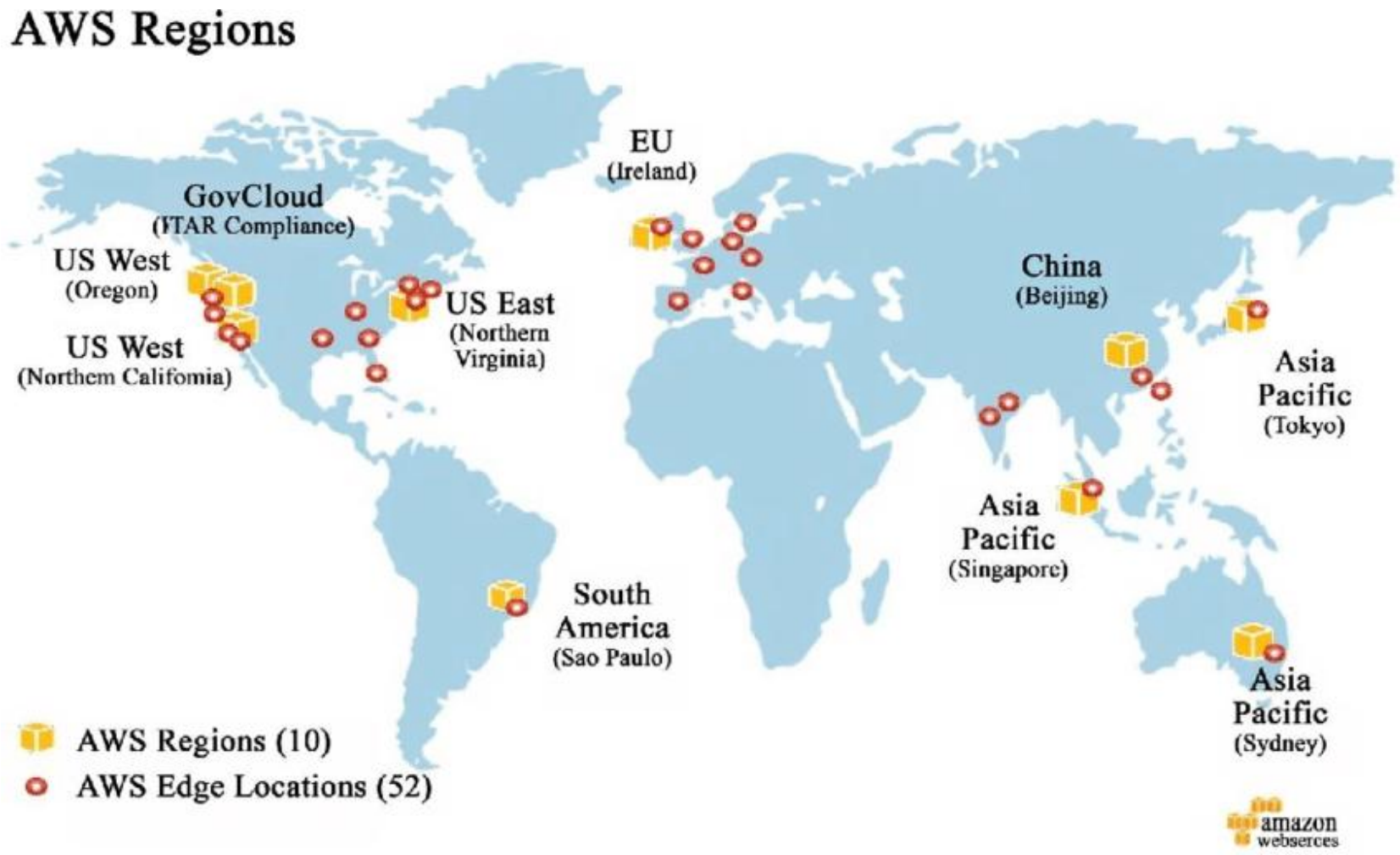
Localizaciones de la nube de Google

https://cloud.google.com/about/locations?hl=es#lightbox-regions-map



Localizaciones de la nube de Amazon

https://www.researchgate.net/figure/Map-showing-the-location-of-Amazon-cloud-datacenters_fig1_301309025



Cloud personal

Cloud personal

- La nube no depende forzosamente de grandes proveedores
  - Lo esencial es el modelo de acceso a recursos compartidos a través de la red, no quién los ofrece
- Un cloud casero es una implementación de cloud computing a pequeña escala
  - Creada y gestionada por una persona o una organización para su propio uso

Cloud personal

- Los recursos físicos (servidores, discos, redes) están en un hogar o en una oficina local
  - Se accede a ellos remotamente, igual que a un servicio en la nube comercial
  - Para implementarlos, se dispone de herramientas software como Nextcloud, Proxmox, ownCloud

Cloud personal

- Ventajas
  - Control total sobre los datos y la infraestructura
  - Independencia de las grandes compañías
- Inconvenientes
  - Requiere mantenimiento propio
  - Incremento del coste, según los recursos de que se quiera disponer