Cloud Computing para Inteligencia Artificial - Guía Completa de Estudio

Tabla de Contenidos

SESIÓN 1: La Sinergia Fundamental

Cloud Computing: Definición y Evolución

Definición NIST (National Institute of Standards and Technology)

Cloud Computing es "un modelo para habilitar el acceso de red ubicuo, conveniente y bajo demanda a un conjunto compartido de recursos informáticos configurables que se pueden aprovisionar y liberar rápidamente con un mínimo esfuerzo de gestión o interacción con el proveedor de servicios".

Evolución Histórica:

- **Años 50-70**: Mainframes IBM - Computación centralizada y compartida
- **Años 90**: Virtualización - VMware permite abstraer hardware del software
- **2006**: Nacimiento de AWS - Amazon crea AWS con S3 y EC2
- **2006-2008**: Google App Engine (PaaS) y Microsoft Azure (Ray Ozzie)

Pioneros de la Idea:

- **John McCarthy (MIT, 1961)**: Predijo que "la computación algún día podría organizarse como un servicio público"
- **J.C.R. Licklider (ARPANET, 1960s)**: Impulsor del concepto de "Intergalactic Computer Network"

Los 5 Principios Fundamentales del NIST

1. Autoservicio Bajo Demanda

- El usuario aprovisiona recursos sin intervención humana
- Antes: Proceso de semanas a través de solicitudes formales
- Ahora: En minutos desde la consola web

2. Amplio Acceso a la Red

- Los servicios están disponibles sobre la red a través de mecanismos estándar
- APIs REST, web, CLI
- Permite automatización mediante Infrastructure as Code

3. Agrupación de Recursos (Multi-tenancy)

- El proveedor agrupa sus recursos para servir a múltiples clientes
- Datos y VMs están lógicamente aislados pero comparten hardware físico
- Permite economías de escala masivas

4. Rápida Elasticidad

- **Escalado Vertical**: Aumentar la potencia de una instancia (más CPU, RAM)
- **Escalado Horizontal**: Añadir más instancias
- La nube es especialmente fuerte en escalado horizontal automático

5. Servicio Medido (Pay-as-you-go)

- Modelo de "utility computing" como la electricidad
- No construyes tu propia central; te conectas a la red
- Pagas solo por lo que usas:
    - AWS EC2: Por segundo/hora
    - AWS S3: Por GB almacenado
    - AWS Lambda: Por milisegundo de ejecución

6. Resiliencia (Bonus)

- Capacidad de resistir fallos y recuperarse rápidamente
- Redundancia en múltiples Zonas de Disponibilidad
- Tolerancia a fallos automática

De CAPEX a OPEX: Un Cambio Fundamental

CAPEX (Capital Expenditure)

- Gasto de capital: Compra de activos físicos (servidores, routers, edificios)
- Gran inversión inicial
- Riesgo de sobreaprovisionar o infraprovisionar

OPEX (Operational Expenditure)

- Gasto operativo: Gastos del día a día (electricidad, salarios, servicios cloud)
- El cloud convierte el enorme CAPEX en OPEX predecible y variable
- **Impacto**: Democratiza el acceso a infraestructura de nivel mundial
    - Una startup puede usar la misma tecnología que Netflix desde el primer día

Modelo de Responsabilidad Compartida

La seguridad en la nube es un acuerdo compartido entre proveedor y cliente.

AWS es responsable de la seguridad DEL cloud:

- Infraestructura física
- Centros de datos seguros
- Red troncal
- Hipervisor de virtualización
- Hardware

TÚ eres responsable de la seguridad EN la nube:

- Tus datos
- Configuración del sistema operativo
- Gestión de usuarios
- Configuración de firewalls
- Cifrado de datos
- Seguridad de la aplicación

La línea de responsabilidad cambia según el modelo de servicio:

Aspecto	IaaS (EC2)	PaaS (Lambda, RDS)	SaaS (Gmail, SageMaker)
Proveedor	Infraestructura, Hipervisor	SO, Middleware, Runtime	Casi todo

Aspecto	IaaS (EC2)	PaaS (Lambda, RDS)	SaaS (Gmail, SageMaker)
Cliente	TODO lo demás	Código, Datos, Config	Gestión de usuarios, Datos

Modelos de Aprovisionamiento de Servicios Cloud

1. IaaS (Infrastructure as a Service)

- **Provisión**: Máquinas virtuales, redes, almacenamiento
- **Control**: El cliente gestiona SO, aplicaciones, datos
- **Ejemplos**: AWS EC2, Microsoft Azure VMs, Google Compute Engine
- **Máxima flexibilidad, máxima responsabilidad**

2. PaaS (Platform as a Service)

- **Provisión**: Plataforma de desarrollo, bases de datos gestionadas
- **Control**: El cliente gestiona código y datos
- **Ejemplos**: AWS Lambda, Heroku, Google App Engine
- **Menos configuración, más rapidez**

3. SaaS (Software as a Service)

- **Provisión**: Aplicaciones completas listas para usar
- **Control**: El cliente solo gestiona usuarios y datos dentro de la app
- **Ejemplos**: Gmail, Salesforce, Microsoft Office 365
- **Máxima facilidad, menor control**

Sinergia Cloud e IA

¿Por qué Cloud Computing es esencial para la IA moderna?

1. **Escalabilidad Masiva de Datos**
    - Los modelos de IA requieren entrenar con billones de datos
    - La infraestructura distribuida del cloud lo permite
2. **Acceso a Hardware Especializado**
    - GPUs (NVIDIA A100, H100) costosas
    - TPUs (Google)
    - FPGAs
    - Sin cloud, estos recursos estarían fuera del alcance de startups
3. **Elasticidad para Experimentación**

- Lanzar 100 experimentos en paralelo
- Pagar solo por lo que usas
- Rapidez de iteración sin inversión inicial

4. **MLOps y DevOps**
    - Automatización de pipelines de ML
    - Versionado de modelos
    - Monitorización en producción
    - Sin cloud, esto requeriría un equipo de infraestructura masivo
5. **Democratización**
    - Un investigador en una universidad rural puede acceder a la misma tecnología que Google

---

SESIÓN 2: Conceptos Clave de Infraestructura Cloud

La Infraestructura Global del Cloud

Los proveedores de cloud han construido una red masiva y distribuida de centros de datos por todo el mundo.

Analogía: Como una red logística global. Si entregas un "paquete" (datos/aplicación) a un usuario en Japón, no lo envías desde Madrid; usas un centro de distribución local en Asia.

Regiones Cloud

Definición: Área geográfica física y separada en el mundo donde se agrupan centros de datos.

Ejemplos de AWS:

- `eu-west-1` : Irlanda
- `us-east-1` : N. Virginia
- `ap-southeast-2` : Sídney

Criterios para su establecimiento:

1. **Latencia**: Estar cerca de los usuarios finales para reducir tiempo de respuesta
2. **Soberanía de datos**: Leyes como GDPR exigen que datos de ciudadanos EU residan en la UE
3. **Disponibilidad de recursos**: Acceso a energía, conectividad, estabilidad geopolítica

Zonas de Disponibilidad (AZs)

Definición: Uno o más centros de datos discretos con energía, refrigeración y redes redundantes dentro de una Región.

Aislamiento Físico:

- Suficientemente lejos para no verse afectadas por desastres únicos (incendios, inundaciones)
- Suficientemente cerca para latencia muy baja entre ellas (<10ms)

Alta Disponibilidad:

- Si una AZ falla, tu aplicación puede seguir funcionando en otra AZ de la misma región
- **Ejemplo**: Entrenas modelo en AZ eu-central-1a, despliegas instancias en 1b y 1c detrás de balanceador de carga

Estructura en AWS:

- Una región típica tiene 2-4 AZs
- Las AZs están numeradas (1a, 1b, 1c)

Edge Locations / Points of Presence (PoPs)

Definición: Red de centros de datos mucho más pequeños y numerosos distribuidos en principales ciudades del mundo.

Propósito Principal: Acercar contenido al usuario final para minimizar latencia.

NO se usan para:

- Ejecutar aplicaciones principales
- Entrenar modelos de IA

Servicios que los utilizan:

1. **Amazon CloudFront (CDN)**
    - Almacena en caché copias de contenido
    - Usuario en Madrid recibe contenido desde PoP en Madrid, no desde Irlanda
2. **Amazon Route 53 (DNS)**
    - Resuelve peticiones de DNS desde PoP más cercano
    - Acelera la conexión inicial

Alcance de los Servicios: Global vs. Regional vs. Zonal

Alcance	Definición	Ejemplos

Alcance	Definición	Ejemplos
Global	Operan sin seleccionar región, únicos en tu cuenta	IAM, Route 53, CloudFront
Regional	Se despliegan en región específica, automáticamente en múltiples AZs	S3, DynamoDB, RDS
Zonal	Se despliegan en AZ específica, eres responsable de redundancia	EC2 instances

Modelo de Responsabilidad Compartida (Repaso)

Para Arquitecturas de IA:

1. **Datos de Entrenamiento**
    - TÚ: Cifrar datasets en S3, controlar acceso con IAM, cumplir GDPR
    - AWS: Asegurar que el disco físico no falle
2. **Modelos Entrenados**
    - TÚ: Proteger (model.pth, saved_model.pb) contra acceso no autorizado
    - AWS: Asegurar durabilidad de S3
3. **Entorno de Cómputo (EC2/SageMaker)**
    - TÚ: Parchear SO, configurar Security Groups, gestionar credenciales
    - AWS: Asegurar que el servidor no se caiga
4. **Endpoints de Inferencia**
    - TÚ: Autenticación/autorización de API
    - AWS: Asegurar disponibilidad del servidor

Identity and Access Management (IAM)

Concepto: Sistema de autenticación y autorización centralizado para AWS.

Componentes:

1. **Usuarios**: Identidades individuales
2. **Grupos**: Colecciones de usuarios
3. **Roles**: Conjuntos de permisos que pueden ser asumidos
4. **Políticas**: Documentos JSON que especifican qué acciones se permiten

Principio de Mínimo Privilegio:

- Dar a cada usuario/rol el mínimo de permisos necesarios
- NO dar permisos "por si acaso"

Multi-Factor Authentication (MFA):

- SIEMPRE activar MFA para cuentas administrativas
- Añade capa de seguridad además de contraseña

Mejores Prácticas:

1. NUNCA uses el usuario root excepto para tareas específicas
2. Crea usuarios administradores desde IAM
3. Usa roles en lugar de claves de acceso cuando sea posible
4. Activa CloudTrail para auditoría

Redes Virtuales (VPC)

Definición: Virtual Private Cloud es tu porción de red privada, lógicamente aislada, dentro de la nube de un proveedor.

Base de tu arquitectura: Es el plano de tu propia oficina en un rascacielos gigante. Tú decides dónde van los muros, las puertas y quién tiene acceso.

Componentes Clave:

- **Región**: La VPC vive en una única región
- **Bloque CIDR**: Define rango de direcciones IP privadas (ej. 10.0.0.0/16)
- **La elección del CIDR es crítica y casi inmutable**

Equivalentes en otros proveedores:

- AWS: VPC (Virtual Private Cloud)
- Azure: VNet (Virtual Network)
- GCP: VPC Network

Subredes Públicas vs. Privadas

Subred Pública:

- Tabla de rutas tiene ruta directa a Internet Gateway (IGW)
- Componentes típicos: Balanceadores, servidores web, bastiones
- Caso de uso IA: Endpoint de API Gateway que expone modelo de inferencia

Subred Privada:

- No tiene ruta directa a internet

- Acceso a internet a través de NAT Gateway
- Componentes típicos: Servidores de aplicación, bases de datos, clústeres de cómputo
- Caso de uso IA: Clúster de instancias EC2 con GPUs procesando datasets sensibles

Gateways: El Acceso al Exterior

Internet Gateway (IGW):

- Componente gestionado, redundante, altamente disponible
- Permite comunicación entre instancias en VPC e internet
- Se asocia a VPC y se añade ruta en tabla de rutas de subred pública (0.0.0.0/0 -> igw-id)

NAT Gateway:

- Permite instancias en subred privada iniciar tráfico hacia internet
- IMPIDE que internet inicie conexiones con esas instancias
- Servicio gestionado: Altamente disponible, escalable
- NAT Instance (legacy): NO recomendado (punto único de fallo)

Security Groups vs. Network ACLs

Security Group (SG):

- Firewall virtual **stateful** a nivel de interfaz de red
- **Stateful**: Si permites tráfico saliente por puerto, respuesta regresa automáticamente
- Solo reglas de permiso (allow)
- Orden no importa, todas se evalúan
- Ámbito: A nivel de instancia
- Uso principal: "Guardia de seguridad personal" para cada instancia

Network ACL (NACL):

- Firewall virtual **stateless** a nivel de subred
- **Stateless**: Debes permitir explícitamente tráfico de respuesta
- Reglas de permiso (allow) y denegación (deny)
- Evaluadas por orden numérico
- Existe regla final implícita (*) que deniega todo
- Uso principal: "Valla perimetral" de la subred

Conclusión Clave: No es uno u otro, usa ambos:

- NACLs para defensa perimetral amplia
- SGs para control de acceso granular y específico

Ejemplo de Arquitectura Correcta (Mínimo Privilegio)

Escenario: Aplicación IA de 3 capas (front-end web React, API de inferencia Flask/FastAPI en ECS, PostgreSQL en RDS)

Opción A (INCORRECTA):

```
Un único SG para todos los recursos:
- TCP 443 desde Internet
- TCP 5000 (API) desde Internet
- TCP 5432 (RDS) desde Internet
```

Opción B (CORRECTA):

```
- SG-WEB: Permite TCP 443 desde 0.0.0.0/0
- SG-API: Permite TCP 5000 solo desde referencia del SG-WEB
- SG-RDS: Permite TCP 5432 solo desde referencia del SG-API
```

La Opción B sigue el **Principio de Mínimo Privilegio** y es la más segura.

VPC Endpoints

Problema: Para que instancia en subred privada acceda a S3, tráfico debe ir a través del NAT Gateway, salir a internet y volver. Es ineficiente y costoso.

Solución: VPC Endpoints permiten conectar VPC a servicios AWS sin IGW o NAT Gateway, manteniendo todo el tráfico dentro de la red privada segura de AWS.

Ventajas:

- Seguridad mejorada: Tráfico no atraviesa internet pública
- Fiabilidad: Menos dependencia de componentes externos
- Coste: Ahorro en costes de procesamiento de datos del NAT Gateway

Tipos:

1. **Gateway Endpoints** (S3, DynamoDB)
    - "Puerta de enlace" especificada como destino en tabla de rutas
    - Sin coste adicional
2. **Interface Endpoints** (AWS PrivateLink)
    - Crean ENI (Elastic Network Interface) con IP privada en subred
    - Para mayoría de servicios AWS (SQS, Kinesis, SageMaker, API Gateway)
    - Con coste por hora y por GB

SESIÓN 3: Servicios de Computación I - Máquinas Virtuales

¿Qué es una Máquina Virtual en la Nube?

Definición: Emulación por software de un sistema informático completo. En la nube, es tu propio servidor privado virtualizado.

Abstracción del Hardware: La VM te proporciona recursos virtualizados (vCPU, RAM, Disco, Red) que son una porción del hardware físico subyacente de un centro de datos.

Rol del Hipervisor: Software clave que crea y gestiona las VMs. Se ejecuta en servidor físico (host) y asigna recursos a cada VM (guest).

Tipos de Hipervisores:

- **Tipo 1 (Bare-metal)**: Xen, KVM, Hyper-V - Se ejecutan directamente sobre hardware

Ventajas Clave de las VMs en la Nube

1. **Aislamiento (Isolation)**
    - Cada VM está lógicamente aislada
    - Un fallo o brecha de seguridad en una VM no afecta directamente a otras
2. **Flexibilidad (Flexibility)**
    - Control total sobre SO, aplicaciones, librerías, configuración
    - "Lienzo en blanco"
3. **Control a Nivel de SO**
    - Acceso de administrador (root)
    - Ajustar kernel, gestionar parches, configurar a bajo nivel
4. **Portabilidad**
    - Imágenes de VMs pueden migrarse entre hosts o incluso entre nubes

Introducción a AWS EC2

EC2 (Elastic Compute Cloud): Servicio de AWS para proporcionar capacidad de cómputo segura y redimensionable en la nube.

Terminología: AWS usa "Instancia" para referirse a una Máquina Virtual.

Clave de EC2: Enorme variedad de "familias" y "tamaños" de instancias, cada una optimizada para diferentes cargas de trabajo.

Sintaxis del Nombre: `familia + generación + atributos_adicionales . tamaño`

Ejemplos:

- `t3.large` : Familia T, generación 3, tamaño large
- `m5.xlarge` : Familia M, generación 5, tamaño xlarge
- `p4d.24xlarge` : Familia P, generación 4d, tamaño 24xlarge

Familias de Instancias

1. Propósito General (M, T)

Serie M (ej. m5, m6g):

- Equilibrio entre CPU, memoria (RAM) y red
- Relación balanceada de vCPU:RAM (típicamente 1:4)
- El pilar para amplia variedad de aplicaciones

Serie T (ej. t2, t3, t4g):

- Instancias de rendimiento ampliable (burstable)
- Línea base de rendimiento de CPU
- Acumulan "créditos de CPU" cuando están inactivas
- Ideal para tráfico variable

Casos de Uso Típicos:

- Servidores web y de aplicaciones
- Entornos de desarrollo y pruebas
- Bases de datos pequeñas y medianas
- Microservicios

2. Optimizadas para Cómputo (C)

Serie C (ej. c5, c6g):

- Priorizan potencia de procesamiento
- Alta relación de vCPU por GB de RAM
- Procesadores más rápidos

Casos de Uso en IA:

- Procesamiento por lotes (Batch Processing)
- Codificación de vídeo (Transcoding)
- Modelado científico y HPC
- Inferencia de Machine Learning (si está muy optimizada para CPU)

3. Optimizadas para Memoria (R, X, Z)

Serie R (ej. r5, r6g):

- Para aplicaciones con uso intensivo de memoria
- Gran cantidad de RAM por vCPU

Serie X (ej. x1, x2g):

- Mayores proporciones de RAM por vCPU
- Pueden llegar a terabytes de RAM

Serie Z (ej. z1d):

- Frecuencia de CPU muy alta + gran cantidad de memoria

Casos de Uso en IA:

- Bases de datos en memoria (SAP HANA, Redis, Memcached)
- Análisis y minería de datos a gran escala
- Pre-procesamiento de grandes datasets para entrenamiento

4. Optimizadas para Almacenamiento (I, D)

Serie I (ej. i3, i4i):

- Alto número de operaciones de entrada/salida (IOPS)
- Almacenamiento local NVMe SSD

Serie D (ej. d2, d3):

- Almacenamiento denso
- Terabytes de almacenamiento local en HDD

Casos de Uso en IA:

- Bases de datos NoSQL (Cassandra, ScyllaDB)
- Data Warehousing a gran escala
- Sistemas de ficheros distribuidos (HDFS, Hadoop)
- Cargas de trabajo con muchos datos temporales

5. Cómputo Acelerado (¡CLAVE PARA IA!) (P, G, Inf, Trn)

Serie P (ej. p3, p4d):

- Equipadas con GPUs NVIDIA de alta gama (A100)
- El estándar para entrenamiento de Deep Learning

Serie G (ej. g4, g5):

- GPUs más versátiles
- Buenas para inferencia de ML, renderizado gráficos, transcodificación

Serie Inf (ej. Inf1, Inf2):

- Chip AWS Inferentia
- Diseñado a medida para inferencia de ML de alto rendimiento y bajo coste

Serie Trn (ej. Trn1):

- Chip AWS Trainium
- Diseñado a medida para entrenamiento de Deep Learning a gran escala

¿Cómo Elegir la Instancia Correcta?

Debes balancear:

1. **vCPU y Arquitectura**
    - ¿Necesitas Intel, AMD o ARM (Graviton)?
    - ¿Cuántos cores?
2. **Memoria (RAM)**
    - ¿Tu aplicación necesita cargar grandes datasets?
3. **Almacenamiento**
    - ¿NVMe ultra-rápido local o almacenamiento persistente en red (EBS)?
4. **Rendimiento de Red**
    - ¿Sensible a latencia?
    - ¿Necesita gran ancho de banda?
5. **Aceleradores de Hardware**
    - ¿La carga se beneficia de GPUs o chips específicos de ML?

6. **Coste**
    - Siempre un factor determinante

Amazon Machine Images (AMIs)

Definición: Plantilla preconfigurada que contiene todo lo necesario para lanzar una instancia.

Contenido de una AMI:

1. Sistema Operativo (Linux, Windows Server)
2. Software de aplicación y librerías (servidor web, stack de Python con TensorFlow, CUDA drivers)
3. Configuraciones de permisos y almacenamiento

Importancia Estratégica:

- **Estandarización**: Cada instancia lanzada es idéntica
- **Despliegue Rápido**: Reduce drásticamente tiempo de configuración
- **En minutos en lugar de horas**

Tipos de AMIs:

1. **AMIs Públicas (Quick Start)**
    - Proporcionadas por AWS con SOs comunes
    - Amazon Linux, Ubuntu, Windows
    - Base más habitual
2. **AMIs del Marketplace**
    - Ofrecidas por terceros (vendedores de software)
    - "Listas para usar" con software preinstalado
    - Ejemplo: AMI con Oracle DB, Fortinet firewall, NVIDIA Deep Learning
3. **AMIs Personalizadas (My AMIs)**
    - Las que tú creas
    - Lanzas una instancia, la configuras exactamente como necesitas
    - Creas una AMI a partir de ella
    - **MEJOR PRÁCTICA para tus aplicaciones**

Almacenamiento para Instancias

1. Almacenamiento de Instancia (Ephemeral)

¿Qué es?: Discos locales conectados físicamente al servidor host que aloja tu instancia.

Ventajas:

- Rendimiento extremo: Latencia muy baja, IOPS altísimo
- Ideal para datos temporales

¡ADVERTENCIA! - Datos Volátiles:

- Los datos se pierden permanentemente si instancia se detiene/termina
- También pueden perderse por fallo de hardware subyacente

Caso de Uso: Espacio para swap, cachés, datos temporales que se generan y eliminan durante un proceso

2. Amazon Elastic Block Store (EBS)

¿Qué es?: Volúmenes de almacenamiento en bloque basados en red, diseñados para alta disponibilidad y durabilidad.

Características Clave:

- **Persistente**: Datos persisten independientemente del ciclo de vida de la instancia
- **Flexible**: Elige tipo de volumen según necesidades (HDD, SSD general purpose, SSD IOPS provisionadas)
- **Backups (Snapshots)**: Tomar instantáneas y guardarlas en S3
- **Acoplable/Desacoplable**: Puedes desacoplar de una instancia y acoplar a otra

Redes para VMs: Direcciones IP

Cada instancia se lanza dentro de una VPC.

Dirección IP Privada:

- Asignada automáticamente desde rango de la red
- Comunicación entre instancias dentro de la misma VPC
- NO accesible desde Internet
- Fija mientras instancia está en ejecución

Dirección IP Pública:

- Para comunicación con Internet
- Por defecto, es dinámica: si detienes e inicias, cambiará

IP Elástica (Elastic IP):

- Dirección IP pública estática
- Puedes asignar a tu cuenta y asociarla a una instancia

- Permanece fija hasta que la liberes explícitamente

Seguridad para VMs: Security Groups

¿Qué son?

- Firewall virtual a nivel de instancia (stateful)
- Controlan tráfico entrante (inbound) y saliente (outbound)

¿Cómo funcionan?

- Se definen reglas que especifican:
  - Protocolo (TCP, UDP, ICMP)
  - Rango de puertos
  - Origen (para inbound) o destino (para outbound)
  - Origen/destino puede ser: IP, rango CIDR, otro Security Group

Stateful: Si permites tráfico entrante en puerto, respuesta saliente se permite automáticamente

Ejemplo para Servidor Web:

```
Regla Inbound: Permitir TCP puerto 80 desde 0.0.0.0/0
Regla Inbound: Permitir TCP puerto 22 (SSH) solo desde IP de tu oficina
```

Acceso Seguro a Instancias: Key Pairs

Para Instancias Linux:

- Se usa Par de Claves basado en criptografía clave pública-privada
- Al crear par: AWS genera clave pública (guardada en instancia) y privada (descargas)
- Para conectar vía SSH:

  ```
  ssh -i mi-clave-privada.pem ec2-user@<ip-publica-instancia>
  ```

Para Instancias Windows:

- Se usa el mismo par de claves para obtener contraseña administrador inicial
- Conectarse vía RDP (Remote Desktop Protocol)

Modelos de Precios de EC2

1. On-Demand (Pago por Uso)

Concepto: Pagas por segundo/hora de uso, sin compromisos.

Pros:

- Máxima flexibilidad
- Sin inversión inicial
- Lanza y termina cuando quieras

Contras:

- El más caro por hora

Ideal para:

- Aplicaciones con cargas irregulares a corto plazo
- Desarrollo y pruebas
- Cuando lanzas aplicación por primera vez y no conoces demanda

2. Reserved Instances (RIs)

Concepto: Te comprometes a usar cierta instancia durante 1 o 3 años a cambio de gran descuento (hasta 72%).

Tipos:

- **Standard RIs**: Mayor descuento, pero ligado a familia de instancia específica en región
- **Convertible RIs**: Menor descuento, permite cambiar familia de instancia, SO

Ideal para:

- Cargas de trabajo estables y predecibles
- Largo plazo (1-3 años)

3. Savings Plans

Concepto: Te comprometes a cierto gasto por hora durante 1 o 3 años a cambio de descuento.

Tipos:

- **EC2 Instance Savings Plans**: Compromisos de gasto por hora en familia de instancias en región
- **Compute Savings Plans**: MÁS flexible, aplica a EC2, Fargate, Lambda en cualquier región

4. Spot Instances

Concepto: Capacidad de cómputo no utilizada que AWS vende con descuento hasta 90%.

El "pero": AWS puede reclamar la instancia con 2 minutos de preaviso.

Ideal para:

- Entrenamientos de ML (toleran interrupciones, soportan checkpoints)

- Procesamiento por lotes
- Tolerante a fallos

SESIÓN 4: Servicios de Computación II - Contenedores y Serverless

¿Por Qué Necesitamos Más que una VM?

Problema Clásico: Un modelo de scikit-learn entrenado con Python 3.8 y Pandas 1.5.1 funciona perfectamente en tu portátil. Al desplegarlo en EC2 con Python 3.9 y Pandas 2.0, ¡falla por incompatibilidades!

Desafíos:

1. **Consistencia de Entornos**: Garantizar que dev, test, prod sean idénticos
2. **Gestión de Dependencias**: Evitar "infierno de las dependencias"
3. **Eficiencia de Recursos**: VMs son pesadas, desperdician RAM y CPU
4. **Portabilidad**: ¿Cómo movemos cargas de trabajo entre nubes?

Virtualización vs. Contenerización

Máquinas Virtuales

- Virtualizan el **hardware**
- Cada VM tiene su propio SO completo (Guest OS)
- **Pesadas**: Ocupan gigabytes
- **Lentas**: Tardan minutos en arrancar
- **Aislamiento Fuerte**: Nivel de hardware

Contenedores

- Virtualizan el **Sistema Operativo**
- Comparten el kernel del SO del host
- Contienen solo la aplicación y sus dependencias
- **Ligeros**: Ocupan megabytes
- **Rápidos**: Arrancan en segundos o milisegundos
- **Aislamiento Suficiente**: A nivel de procesos (namespaces, cgroups en Linux)

Docker: El Estándar de Facto

Docker es plataforma open-source para construir, distribuir y ejecutar contenedores.

Componentes Clave:

1. **Imagen Docker (Image)**
    - Plantilla inmutable de solo lectura
    - Se construye por capas (Ubuntu → Python → dependencias → código)
2. **Dockerfile**
    - Fichero de texto que define, paso a paso, cómo construir imagen
    - "Infraestructura como código" para tu aplicación
3. **Contenedor Docker**
    - Instancia en ejecución de una imagen
    - Proceso aislado y ejecutable
4. **Registros (Registries)**
    - Repositorio para almacenar y distribuir imágenes
    - Docker Hub (público)
    - AWS ECR, GCP Artifact Registry, Azure CR (privados)

Anatomía de un Dockerfile para IA:

```
# 1. Usar imagen base oficial con Python
FROM python:3.9-slim

# 2. Establecer directorio de trabajo
WORKDIR /app

# 3. Copiar fichero de dependencias
COPY requirements.txt .

# 4. Instalar dependencias
RUN pip install --no-cache-dir -r requirements.txt

# 5. Copiar código de la aplicación
COPY . .

# 6. Exponer puerto
EXPOSE 5000
```

```
# 7. Comando para ejecutar
CMD ["python", "app.py"]
```

Comandos básicos:

```
docker build -t mi-api-ia .                    # Construir imagen
docker run -p 5000:5000 mi-api-ia              # Ejecutar contenedor
```

Beneficios de Contenedores para IA/ML

1. **Reproducibilidad de Experimentos**
    - Dockerfile fija versión del SO, Python, librerías
    - Cualquier miembro del equipo reproduce exacto con `docker run`
2. **Encapsulación de Modelos**
    - Modelo entrenado (.h5, .pkl) + todo su entorno de ejecución
    - Artefacto portable y versionable
3. **Despliegue Consistente**
    - Mismo contenedor probado en local se despliega en producción
    - Elimina sorpresas
4. **Aislamiento de Dependencias**
    - Diferentes versiones de CUDA en proyectos distintos
    - Perfectamente aisladas

El Desafío de la Escala

Ejecutar un contenedor es fácil. Pero ¿qué cuando tienes múltiples microservicios en contenedores y miles de usuarios?

Preguntas a resolver:

1. **Despliegue y Escalado**: ¿Cómo lanzo 10 réplicas? ¿Y si necesito 50?
2. **Alta Disponibilidad**: ¿Qué si el servidor donde corre contenedor se cae?
3. **Balanceo de Carga**: ¿Cómo distribuyo peticiones entre réplicas?
4. **Descubrimiento de Servicios**: ¿Cómo microservicio A encuentra a B si IPs son dinámicas?
5. **Actualizaciones sin Caídas**: ¿Cómo actualizo modelo sin interrumpir servicio?

Solución: Un **orquestador** - Kubernetes

Kubernetes (K8s): El Sistema Operativo del Cloud

Definición: Plataforma open-source que automatiza despliegue, escalado y gestión de aplicaciones en contenedores.

Ventajas:

- **Portabilidad**: Funciona en AWS, GCP, Azure, on-premise - **Evita vendor lock-in**
- **Gran Ecosistema**: Enorme comunidad, miles de herramientas se integran
- **Auto-reparación**: Si contenedor falla, K8s lo reinicia. Si nodo muere, K8s mueve contenedores a nodos sanos

Filosofía: Ofrece capa de abstracción sobre infraestructura, permitiendo desarrolladores de IA centrarse en modelos en lugar de detalles de servidores.

Conceptos Clave de Kubernetes

Cluster

- Conjunto de todas las máquinas (nodos) que gestiona Kubernetes

Nodo (Node)

- Una máquina, física o virtual (ej: instancia EC2)
- **Control Plane**: El cerebro del cluster, toma decisiones globales
- **Worker Node**: Ejecuta los contenedores

Pod

- Unidad de despliegue más pequeña
- Wrapper alrededor de uno o más contenedores
- Generalmente: 1 Pod = 1 contenedor
- Los pods son efímeros

Deployment

- Declara el estado deseado
- Ejemplo: "Quiero 3 réplicas de mi-api-ia v1.2"
- Se encarga de crear los Pods y mantenerlos

Service

- Proporciona IP y DNS estables para un conjunto de Pods
- Permite comunicación entre servicios

- Actúa como balanceador de carga interno

¿Gestionar K8s o Usarlo?

Instalar y mantener un cluster de Kubernetes es **muy complejo**. Requiere expertos en redes, seguridad, sistemas.

Dilema: ¿Queremos ser expertos en Kubernetes o en construir modelos de IA?

Solución: Servicios gestionados de contenedores donde **el proveedor gestiona el Control Plane** y nosotros solo desplegamos.

Amazon ECS (Elastic Container Service)

Orquestador propietario de AWS.

Conceptos Clave:

- **Task Definition**: JSON blueprint con imagen Docker, CPU/Memoria, puertos, etc. (Similar a docker-compose)
- **Task**: Instancia en ejecución de una Task Definition (Similar a Pod en K8s)
- **Service**: Mantiene número deseado de Tasks y se integra con balanceadores de carga (ALB)
- **Cluster**: Agrupamiento lógico donde se ejecutan las Tasks

Ventaja Principal: Mucho más simple si ya estás en ecosistema AWS. Integración nativa y profunda con otros servicios.

ECS: Modos de Lanzamiento

Modo EC2

- TÚ provisionas y gestionas cluster de instancias EC2
- TÚ eres responsable de parchear, escalar, securizar SO
- Control total: Elige tipo de instancia (GPUs), usa EBS, etc.
- Pago: Pagas por instancias EC2 24/7, independientemente de tráfico

Modo Fargate (Serverless)

- No gestionas servidores
- AWS provisiona infraestructura "just-in-time"
- Solo defines tu Task y Fargate la ejecuta
- Simplicidad máxima: Olvídate del SO y escalado
- Pago: Solo por vCPU y memoria que tus contenedores consumen mientras se ejecutan

Amazon EKS (Elastic Kubernetes Service)

Servicio de Kubernetes gestionado de AWS.

- AWS gestiona disponibilidad y escalabilidad del Control Plane
- TÚ gestionas Worker Nodes (instancias EC2) o usas Fargate para Pods serverless
- Certificado por CNCF: Kubernetes "puro"
- Aplicaciones K8s estándar correrán en EKS

¿Cuándo elegir EKS?

- Necesitas potencia y ecosistema completo de Kubernetes
- Quieres estrategia multi-cloud o híbrida
- Tu equipo ya tiene experiencia en K8s

Consideraciones de IA: GPUs y Kubeflow

Uso de GPUs:

- Tanto ECS (con EC2) como EKS permiten instancias con GPU
- Cruciales para inferencia de Deep Learning

Kubeflow:

- Proyecto open-source que convierte Kubernetes en plataforma de MLOps
- Características:
  - Pipelines de ML portables y escalables
  - Jupyter Notebooks como servicio
  - KFServing/KServe para servicio de modelos
  - Hyperparameter Tuning
- Se puede instalar sobre EKS para crear potente plataforma de IA

Serverless: Más Allá de Contenedores

Hemos visto "serverless para contenedores" con Fargate. Pero hay abstracción aún mayor: **Functions as a Service (FaaS)**.

Principios Clave del Serverless:

1. **Sin Gestión de Servidores**
   - No provisionas, parcheas ni escalas servidores
   - Ni siquiera contenedores o clusters
2. **Escalado Automático e Instantáneo**

- Plataforma escala de cero a miles de ejecuciones en paralelo

3. **Paga por Ejecución**
    - Si tu código no se ejecuta, no pagas
    - Pagas por milisegundo de ejecución y memoria asignada

Es la computación más orientada a eventos: Tu código "duerme" hasta que algo (un evento) lo despierta.

AWS Lambda: Tu Código en la Nube

Lambda es el servicio FaaS de AWS.

Concepto Central: Escribes una función en lenguaje preferido (Python, Node.js, etc.), la subes a Lambda. Eso es todo.

Triggers (Disparadores):

- Petición HTTP a través de API Gateway
- Subida a S3
- Mensaje en SQS
- Cambio en DynamoDB
- Programada (EventBridge)
- ¡Y más de 200 integraciones en AWS!

Modelo de Ejecución de Lambda

Handler: Punto de entrada de tu código.

```
def handler(event, context):
    # event: datos del evento que disparó la función
    # context: información sobre la invocación
    return {"statusCode": 200, "body": "Hello World"}
```

Entorno de Ejecución: Temporal e aislado con tu código, runtime del lenguaje y dependencias.

Cold Start vs. Warm Start:

- **Cold Start**: Primera invocación o tras inactividad. Lambda crea entorno desde cero. Latencia añadida (ms a segundos)
- **Warm Start**: Invocación rápida tras uso reciente. Lambda reutiliza entorno. Mucho más rápido

Layers: Forma de empaquetar librerías y dependencias (NumPy, Pandas) para compartir entre múltiples funciones.

Lambda para IA: Casos de Uso

Lambda tiene limitaciones (15 min de ejecución, tamaño limitado). NO es para entrenar modelos grandes.

1. **Preprocesamiento de Datos**
    - Trigger: Imagen se sube a S3
    - Lambda: Redimensiona, extrae metadatos, normaliza, guarda lista para entrenamiento
2. **Inferencia en Tiempo Real**
    - Trigger: POST a endpoint API Gateway
    - Lambda: Carga modelo ligero, realiza predicción, devuelve resultado
    - Ejemplo: Regresión logística, árbol de decisión, modelo NLP pequeño
3. **Orquestación de Flujos de ML**
    - AWS Step Functions orquesta pipeline complejo
    - Ejemplo: Función de validación → Función de entrenamiento → Función de despliegue

Serverless para Contenedores: Lo Mejor de Dos Mundos

A veces Lambda no es suficiente (dependencias binarias específicas, runtime personalizado, paquete grande), pero quieres simplicidad.

Solución: Ejecutar contenedores en plataforma serverless

- **AWS Fargate**: Ejecuta contenedores ECS/EKS sin gestionar nodos
- **Google Cloud Run**: Extremadamente popular. Despliegas contenedor, GCP te da endpoint HTTPS. Escala a cero si no hay tráfico
- **Azure Container Instances**: Ejecuta contenedor único rápidamente. Serverless básico para contenedores

---

SESIÓN 5: Servicios de Almacenamiento I

La Tríada del Almacenamiento Cloud

Tres paradigmas fundamentales, cada uno diseñado para propósito específico:

1. **Almacenamiento de Objetos**
    - Para escala masiva y datos no estructurados

- Analogía: Servicio de valet - entregas tu objeto, te dan ID único para recuperarlo

2. **Almacenamiento de Bloques**
    - Almacenamiento de más alto rendimiento para cargas estructuradas
    - Analogía: Disco duro externo virtual que conectas a tu servidor
3. **Almacenamiento de Ficheros**
    - Para acceso compartido y jerarquías de directorios
    - Analogía: Unidad de red compartida (NAS) en empresa

El Factor Clave: Latencia

Aspecto	Bloques (EBS)	Ficheros (EFS)	Objetos (S3)
Ubicación Física	Una AZ	Múltiples AZs	Múltiples AZs
Latencia	Muy Baja	Baja	Mayor (no milisegundos)
Rendimiento	IOPS rápidas	Baja latencia	Throughput masivo
Acoplamiento	Junto a instancia	Red (NFS)	Red (HTTP)

¿Qué es Almacenamiento de Objetos?

Definición: Gestiona datos como "objetos" discretos. NO hay jerarquía de carpetas; es estructura plana (flat namespace).

Componentes de un Objeto:

- **Datos**: El contenido del fichero (imagen, vídeo, CSV, modelo serializado)
- **Identificador Único (Key)**: El "nombre" dentro del contenedor
- **Metadatos**: Información descriptiva (tipo de contenido, fecha creación, etiquetas)

Características Clave:

- Acceso a través de APIs RESTful (HTTP/S), no montando unidad
- Ideal para: Datos no estructurados, backups, data lakes, contenido multimedia
- Escalabilidad: Prácticamente ilimitada, desde KB hasta PB

Introducción a Amazon S3

S3 (Simple Storage Service): Uno de los servicios fundacionales de AWS (2006). Durabilidad del 99.999999999% (once nueves).

Buckets: Contenedores donde se almacenan objetos

- Nombre único globalmente
- Se crean en región específica
- El nombre es global, pero el bucket es regional

Objetos: Los ficheros que subes

- Tamaño máximo por objeto: 5TB
- Se identifican por clave única dentro del bucket
- Pueden tener metadatos personalizados

La "Ilusión" de Carpetas en S3

S3 tiene namespace plano, pero simula estructura de carpetas usando prefijos en la clave (nombre) del objeto.

Ejemplo:

```
proyecto-ia/datasets/imagenes/gato.jpg
proyecto-ia/datasets/imagenes/perro.jpg
```

Aquí `proyecto-ia/datasets/imagenes/` es el prefijo.

Importante:

- La consola interpreta `/` para mostrar vista jerárquica
- Para S3, son simplemente dos objetos con nombres largos
- No puedes "crear una carpeta vacía"
- Una "carpeta" aparece cuando subes el primer objeto con ese prefijo

Clases de Almacenamiento (Tiers) - Acceso Frecuente

S3 Standard

- **Uso**: Acceso muy frecuente, requieren baja latencia y alto rendimiento
- **Durabilidad**: Distribuido en al menos 3 AZs
- **Coste**: Más alto en almacenamiento ($/GB), pero más bajo en acceso ($/petición)
- **Caso IA**: Servir activos de aplicación web, datasets en uso activo para entrenamiento

S3 Intelligent-Tiering

- **Uso**: Datos con patrones de acceso desconocidos o cambiantes
- **Funcionamiento**: Monitoriza acceso, mueve automáticamente entre niveles

- **Coste**: Pequeña tarifa de monitorización, pero potencial ahorro significativo
- **Caso IA**: Datasets de entrenamiento con uso variable en el tiempo

Clases de Almacenamiento - Acceso Poco Frecuente

S3 Standard-IA

- **Uso**: Acceso menos frecuente, pero disponibles rápidamente (milisegundos)
- **Durabilidad**: Mínimo 3 AZs
- **Coste**: Almacenamiento más barato, pero tarifa de recuperación
- **Caso IA**: Backups, logs antiguos, datasets esporádicos

S3 One Zone-IA

- **Uso**: Similar a Standard-IA, pero datos no críticos fácilmente reproducibles
- **Durabilidad**: Una AZ única
- **Coste**: ~20% más barato que Standard-IA
- **Caso IA**: Resultados intermedios de procesamiento, datos generados reproducibles

Clases de Almacenamiento - Archivador Digital (Glacier)

S3 Glacier Instant Retrieval

- **Acceso**: Milisegundos, como S3 Standard-IA
- **Uso**: Archivos raramente, pero recuperación inmediata (ej. archivos médicos)

S3 Glacier Flexible Retrieval

- **Acceso**: Flexible, minutos (acelerado) a horas (bulk)
- **Uso**: Archivado estándar, backups a largo plazo donde retraso es aceptable

S3 Glacier Deep Archive

- **Acceso**: Lento, típicamente 12 horas
- **Uso**: Almacenamiento más barato de AWS, retención regulatoria, datos raramente accedidos

---

SESIÓN 6: Servicios de Almacenamiento II

Almacenamiento de Ficheros (NAS) - Concepto

Definición: Network Attached Storage permite a múltiples clientes (servidores, contenedores) acceder y compartir datos a través de una red.

¿Cómo funciona?: Usa protocolos estándar como NFS (Network File System) para Linux/Unix y SMB/CIFS para Windows.

Característica Clave: Mantiene semántica de sistema de ficheros tradicional: jerarquía de directorios, ficheros, permisos (POSIX, ACLs), bloqueo de ficheros. Como tener unidad de red compartida, pero en la nube.

Casos de Uso de NAS en IA

1. **Entornos de Desarrollo Colaborativos**
    - Varios Data Scientists montan mismo sistema de ficheros en sus instancias
    - Acceso a código fuente, notebooks, datasets de experimentación
2. **Home Directories Centralizados**
    - Directorio /home persistente y compartido para clúster de usuarios
3. **Entrenamiento de ML Distribuido**
    - Múltiples nodos de entrenamiento leen mismo dataset almacenado en sistema de ficheros
4. **Aplicaciones Heredadas**
    - Aplicaciones monolíticas diseñadas para leer/escribir en sistema de ficheros local

AWS File Storage: Amazon EFS

Amazon EFS (Elastic File System): El servicio de NAS gestionado por excelencia en AWS.

Protocolo: NFSv4

Características Principales:

- Totalmente Gestionado y Elástico: No hay que provisionar tamaño, crece/decrece automáticamente
- Altamente Disponible: Datos almacenados redundantes en múltiples AZs
- Clases de Almacenamiento: Standard e Infrequent Access
- Modos de Rendimiento: General Purpose y Max I/O

Integración: Acceso nativo desde EC2, ECS, EKS y AWS Lambda

Precio: Solo pagas por lo que usas, sin necesidad provisionar capacidad por adelantado. Ideal para entornos IA con necesidades cambiantes.

Limitación: No soportado multi-región de forma nativa (se necesitan sistemas de replicación como AWS Datasync)

AWS File Storage: Amazon FSx (Servicios Especializados)

FSx for Windows File Server

- Sistema de ficheros Windows nativo, totalmente gestionado
- Protocolo: SMB
- Caso de Uso: Aplicaciones .NET, SQL Server, cargas Windows

FSx for Lustre

- Sistema de ficheros code abierto optimizado para HPC (High Performance Computing)
- Rendimiento Extremo: Cientos de GB/s throughput, millones de IOPS, sub-milisegundos latencia
- **Caso de Uso en IA**: Entrenamiento Deep Learning a gran escala (datasets grandes de imágenes/vídeos)

Bases de Datos Relacionales (SQL) - Concepto

Modelo Relacional: Organiza datos en tablas (relaciones), filas (tuplas), columnas (atributos).

Características Clave:

- **Estructura Rígida** (Schema-on-write): Esquema se define antes de insertar datos
- **Lenguaje SQL**: Estándar para definir, manipular, consultar
- **Garantías ACID**: Atomicidad, Consistencia, Aislamiento, Durabilidad

Las garantías ACID permiten transacciones seguras incluso con alta concurrencia, fundamental para aplicaciones donde integridad es crítica.

Casos de Uso de SQL en IA

1. **Metadatos de Modelos y Datasets**
    - Nombre del modelo, versión, hiperparámetros
    - Métricas de evaluación (Accuracy, F1-score)
    - Ruta al artefacto del modelo en S3
    - Linaje de los datos
2. **Resultados Estructurados**

- Guardar predicciones del modelo si tienen estructura tabular fija
- Ejemplo: Tabla con ID_cliente, probabilidad_churn, fecha_prediccion

3. **Feature Stores**

   - Muchas implementaciones de Feature Stores usan BBDD relacional
   - Catálogo de features y metadatos asociados

4. **Gestión de Aplicaciones de IA**

   - Información de usuarios, roles, configuraciones
   - Logs de auditoría

AWS RDS (Relational Database Service)

¿Qué es?: Servicio totalmente gestionado que simplifica configuración, operación y escalado de BBDD relacionales en la nube.

Valor Principal: AWS se encarga de tareas pesadas:

- Provisioning de hardware
- Instalación y parcheo del software de BBDD
- Backups automáticos
- Replicación para alta disponibilidad

Motores Soportados:

- Amazon Aurora
- PostgreSQL
- MySQL
- MariaDB
- Oracle Database
- Microsoft SQL Server

Amazon Aurora: BBDD Relacional Re-inventada para el Cloud

¿Qué es?: Motor de BBDD relacional compatible con MySQL y PostgreSQL, desarrollado por AWS para mayor rendimiento, disponibilidad y escalabilidad.

Arquitectura Clave: Separación de cómputo y almacenamiento

- Almacenamiento es una capa de servicio distribuida, auto-reparable, tolerante a fallos
- Volumen de BBDD se replica 6 veces en 3 AZs

Beneficios:

- **Rendimiento**: Hasta 5x rendimiento de MySQL estándar, 3x PostgreSQL estándar
- **Escalabilidad**: Hasta 15 réplicas de lectura de baja latencia, volumen crece automáticamente hasta 128 TB
- **Disponibilidad**: Failover casi instantáneo a réplica
- **Aurora Serverless**: Arranca, apaga y escala automáticamente según carga

NoSQL: Flexibilidad y Escala

Concepto: Bases de datos que NO usan modelo relacional. Ofrecen flexibilidad para datos semi-estructurados.

Tipos Principales:

1. **Clave-Valor** (DynamoDB)
    - Más simple y rápido
    - Perfecto para acceso por clave
2. **Documentales** (DocumentDB, MongoDB)
    - Almacenan documentos JSON/BSON
    - Flexible, semi-estructurado
3. **Grafos** (Neptune)
    - Para relaciones complejas
    - Recomendaciones, redes sociales
4. **En Memoria** (ElastiCache)
    - Redis, Memcached
    - Caché ultra-rápido
    - Para resultados de sesión, contadores

Data Lakes vs. Data Warehouses

Data Lake

- **Concepto**: Repositorio centralizado y flexible que almacena datos estructurados, semi-estructurados y no estructurados a cualquier escala
- **Filosofía**: Schema-on-read (aplicas estructura al leer)
- **Desacoplamiento**: Separa almacenamiento del cómputo - múltiples herramientas sobre mismo repositorio
- **Construcción**: Generalmente sobre S3 (bajo coste)
- **Caso de Uso**: Análisis exploratorio, datos brutos

Data Warehouse

- **Concepto**: Almacenamiento optimizado para análisis, datos ya procesados y estructurados
- **Filosofía**: Schema-on-write (estructura antes de insertar)
- **Acoplamiento**: Almacenamiento y cómputo más acoplados
- **Construcción**: Redshift, Snowflake (optimizados para OLAP)
- **Caso de Uso**: Reportes, BI, análisis estructurados

No es uno u otro: Muchas veces coexisten. Data Lake (S3) para datos brutos → Subconjuntos limpios → Data Warehouse para análisis de alto rendimiento.

AWS Lake Formation: Construyendo tu Data Lake

El Reto del Data Lake: Construir desde cero es complejo (ingesta, limpieza, catalogación, seguridad).

Solución: AWS Lake Formation simplifica enormemente en días en lugar de meses.

Funciones Principales:

- Automatiza ingesta de datos con "blueprints"
- Centraliza catálogo (usando AWS Glue Data Catalog)
- Define y gestiona permisos granulares (nivel BBDD, tabla, incluso columna)

SESIÓN 4: Servicios de Red Avanzados y Conectividad para IA

Revisión y Profundización en VPC

Una VPC es tu porción de red privada, lógicamente aislada, dentro de la nube.

Componentes Clave:

- **Región**: La VPC vive en una única región
- **Bloque CIDR**: Define rango de direcciones IP privadas (ej. 10.0.0.0/16)
- **La elección del CIDR es crítica y casi inmutable**

Equivalentes:

- AWS: Virtual Private Cloud (VPC)
- Azure: Virtual Network (VNet)
- GCP: VPC Network

Diseño de VPC: Subredes Públicas vs. Privadas

Subred Pública

- Tabla de rutas tiene ruta directa a Internet Gateway (IGW)
- Componentes: Balanceadores, servidores web, bastiones
- Caso IA: Endpoint de API Gateway exponiendo modelo de inferencia

Subred Privada

- NO tiene ruta directa a internet
- Acceso a internet a través de NAT Gateway
- Componentes: Servidores de aplicación, bases de datos, clústeres de cómputo
- Caso IA: Clúster de EC2 con GPUs procesando datasets sensibles

Gateways: El Acceso al Exterior

Internet Gateway (IGW)

- Componente gestionado, redundante, altamente disponible
- Permite comunicación entre instancias en VPC e internet
- Se asocia a VPC y se añade ruta en tabla de rutas (0.0.0.0/0 -> igw-id)

NAT Gateway

- Permite instancias en subred privada iniciar tráfico hacia internet
- IMPIDE que internet inicie conexiones con esas instancias
- Servicio gestionado: Altamente disponible, escalable
- NAT Instance (legacy): NO recomendado (punto único de fallo)

Capas de Defensa: Security Groups vs. NACLs

Security Group (SG)

- Firewall virtual **stateful** a nivel de interfaz de red
- Si permites tráfico saliente, respuesta regresa automáticamente
- Solo reglas de permiso (allow)
- Evaluación: Todas las reglas se evalúan, orden no importa
- Ámbito: Instancia
- Uso: "Guardia de seguridad personal"

Network ACL (NACL)

- Firewall virtual **stateless** a nivel de subred
- Debes permitir explícitamente tráfico de respuesta
- Reglas de permiso (allow) y denegación (deny)
- Evaluación: Por orden numérico
- Regla final implícita (*) que deniega todo
- Uso: "Valla perimetral" de subred

Conclusión: Usa ambos - NACLs para defensa amplia, SGs para control granular.

VPC Endpoints

Problema: Para acceder a S3 desde subred privada, tráfico va a través de NAT Gateway, sale a internet, vuelve. Ineficiente y costoso.

Solución: VPC Endpoints - conectan VPC a servicios AWS sin IGW o NAT Gateway.

Tipos:

1. **Gateway Endpoints** (S3, DynamoDB)
    - "Puerta de enlace" en tabla de rutas
    - Sin coste adicional
2. **Interface Endpoints** (PrivateLink)
    - Crean ENI con IP privada en subred
    - Mayoría servicios AWS
    - Con coste

Ventajas:

- Seguridad mejorada: Sin internet pública
- Fiabilidad: Menos dependencia externa
- Coste: Ahorro NAT Gateway

Conectividad Privada: VPC Peering y Transit Gateway

VPC Peering

- Conexión directa entre dos VPCs
- Permite que instancias en ambas VPCs se comuniquen como si estuvieran en la misma red
- No hay acceso a internet entre ellas

Transit Gateway

- Servicio de conectividad central
- Conecta múltiples VPCs y redes on-premise
- Mucho más escalable que peering punto a punto

El Mundo Híbrido: Conectando On-Premise con Cloud

Conexiones VPN

- Virtual Private Network sobre internet público
- Más rápido de configurar
- Bajo coste
- Latencia variable

AWS Direct Connect

- Conexión privada dedicada entre on-premise y AWS
- Rendimiento superior
- Seguridad mejorada
- Mayor coste, pero para cargas críticas vale la pena

Caso IA: Entrenamiento manual en on-premise puede transferir datasets a nube, usar GPU de AWS, traer resultados de vuelta.

Aceleración y Alcance Global: CDNs y DNS Inteligente

Amazon CloudFront (CDN)

- Distribuye contenido desde ubicación cercana a usuario
- Reduce latencia dramáticamente
- Cachea contenido en Edge Locations

Amazon Route 53 (DNS Inteligente)

- Servicio de DNS altamente disponible
- Políticas de enrutamiento:
    - **Simple**: Round-robin
    - **Ponderada**: Porcentaje de tráfico a cada destino
    - **Basada en Latencia**: Usuario recibe de región con menor latencia
    - **Failover**: Activo-pasivo para DR
    - **Geo-proximidad**: Basado en ubicación geográfica

Caso de Uso IA: Modelo de traducción desplegado en múltiples regiones, Route 53 envía usuario a región con menor latencia basándose en su ubicación.

Defensa Activa: DDoS y WAF

AWS Shield

- **Shield Standard**: Automático y gratuito para todos los clientes AWS. Defiende contra ataques DDoS comunes de capa 3 (red) y 4 (transporte)
- **Shield Advanced**: Servicio de pago, protección mejorada para ataques sofisticados. Acceso 24/7 a Equipo de Respuesta DRT.

AWS WAF (Web Application Firewall)

- Firewall para aplicaciones web/APIs
- Capa 7 (Aplicación) del modelo OSI
- Protege contra:
  - Inyección SQL (SQLi)
  - Cross-Site Scripting (XSS)
  - OWASP Top 10
- Se integra con CloudFront, ALB, API Gateway
- Creas reglas basadas en IP, cabeceras HTTP, cuerpo de petición

Arquitectura de Seguridad en Profundidad

Capas:

1. **Edge**: Route 53 (DNS) y CloudFront (CDN)
2. **Protección DDoS**: AWS Shield
3. **Inspección Web**: AWS WAF
4. **Borde de Subred**: Network ACLs
5. **Recurso**: Security Groups

Cada capa protege a la siguiente. Si atacante supera una, se encuentra con la siguiente.

SESIÓN 8: Plataformas de Machine Learning Gestionadas I

El Ciclo de Vida del Machine Learning (MLOps)

MLOps: Conjunto de prácticas que busca desplegar y mantener modelos de ML en producción de manera fiable y eficiente. Aplicación de principios DevOps al ciclo de vida de ML.

No es lineal: Es un ciclo continuo de mejora y adaptación.

Fases del Ciclo MLOps

1. Recolección y Preparación de Datos

- Ingesta desde diversas fuentes (Data Lakes S3, Bases de datos, APIs)
- Limpieza, validación, exploración (EDA)

2. Ingeniería de Características

- Creación de variables predictivas
- Transformaciones, codificaciones (one-hot), normalización
- Paso crucial y a menudo el más costoso en tiempo

3. Desarrollo y Entrenamiento del Modelo

- Experimentación con diferentes algoritmos
- Ajuste de hiperparámetros (Hyperparameter Optimization - HPO)
- Seguimiento de experimentos para reproducibilidad

4. Evaluación y Validación

- Medición con métricas clave (Accuracy, F1-score, RMSE)
- Análisis de sesgos (bias) y equidad (fairness)

5. Despliegue del Modelo

- Empaquetar modelo (Pickle, ONNX)
- Crear endpoint de API para inferencia
- Versionado de modelos

6. Monitorización y Reentrenamiento

- Monitorización de rendimiento
- Detección de Data Drift (cambios en distribución de datos de entrada)
- Detección de Concept Drift (cambios en relación entre variables y salida)
- Reentrenamiento automático cuando se degrada rendimiento

Desafíos del MLOps "Do It Yourself" (DIY)

1. **Fragmentación**

- Herramientas dispares para cada etapa (Jupyter, Scikit-learn, Flask, Cron)

2. **Reproducibilidad**
    - ¿Cómo garantizamos que experimento que funcionó en portátil funcione igual en producción?
3. **Escalabilidad**
    - Entrenar con 10 GB local es fácil. ¿Y con 10 TB? ¿Gestionar clúster de GPUs?
4. **Colaboración**
    - Múltiples roles necesitan trabajar sobre mismos artefactos
5. **Gobernanza**
    - ¿Quién tiene acceso a qué? ¿Auditoría de modelos?

Conclusión: Hacer MLOps desde cero es complejo, costoso y desvía foco de creación de valor con IA.

Ventajas de Plataformas de ML Gestionadas

1. **Abstracción de Infraestructura**
    - Olvídate de configurar servidores, redes, clústeres Kubernetes
    - Pides "máquina de entrenamiento con 4 GPUs", plataforma la provisiona
2. **Herramientas Integradas**
    - Conjunto cohesivo que cubre todo ciclo de vida MLOps
3. **Escalabilidad Bajo Demanda**
    - Pasar de entrenar en CPU a clúster de cientos de GPUs con cambio en configuración
4. **Optimización de Costes**
    - Mecanismos como Spot instances integrados
    - Reducción drástica de costes de entrenamiento
5. **Democratización de MLOps**
    - Facilita implementación de pipelines robustos sin gran equipo de ingenieros de plataforma

El Ecosistema de Plataformas de ML

Amazon SageMaker

- Más maduro y mayor cuota de mercado
- Granularidad y conjunto de herramientas extremadamente amplio

Google Vertex AI

- Plataforma muy unificada
- Fuerte integración con BigQuery
- Enfoque en AutoML muy potente

Azure Machine Learning

- Excelente integración con ecosistema empresarial de Microsoft
- Interfaz de diseñador visual muy intuitiva

¿Qué es Amazon SageMaker?

Plataforma de Machine Learning totalmente gestionada lanzada por AWS en 2017.

Objetivo: Proporcionar a desarrolladores y científicos de datos capacidad de construir, entrenar y desplegar modelos de ML rápidamente y a cualquier escala.

Filosofía: Modular. Puedes usar toda plataforma integrada o solo componentes que necesites.

No es una herramienta: Suite de servicios mapeados a cada etapa del ciclo de vida de ML.

Mapeando SageMaker al Ciclo de MLOps

Preparar Datos

- **SageMaker Studio Notebooks**: IDE basado en web
- **Data Wrangler**: Preparación visual de datos
- **Processing Jobs**: Ejecutar scripts de procesamiento a escala

Construir y Entrenar

- **Feature Store**: Repositorio centralizado de características
- **Algoritmos integrados**: Optimizados para AWS
- **Training Jobs**: Entrenar con frameworks (TensorFlow, PyTorch)
- **Automatic Model Tuning**: Optimización de hiperparámetros
- **Experiments, Debugger**: Seguimiento y depuración

Desplegar y Gestionar

- **SageMaker Endpoints**: API para inferencia en tiempo real
- **Batch Transform**: Inferencia por lotes

- **Model Monitor**: Monitorización en producción
- **Pipelines**: Orquestación de flujos de trabajo

La Base: Almacenamiento con Amazon S3

Casi todas las operaciones en SageMaker leen y escriben datos en S3.

S3 es:

- Almacenamiento de objetos altamente escalable, duradero, seguro
- El "disco duro de la nube" para datos de ML

Flujo Típico:

1. Datos brutos se almacenan en bucket S3
2. Trabajo de preprocesamiento lee desde S3
3. Datos procesados se guardan de nuevo en S3
4. Trabajo de entrenamiento lee datos procesados desde S3
5. Modelo entrenado (artefacto) se guarda en S3

SageMaker Studio: El IDE para Machine Learning

IDE basado en web completamente unificado para Machine Learning.

Componentes en una sola interfaz:

- Notebooks Jupyter gestionados
- Acceso a Data Wrangler
- Seguimiento de Experimentos (SageMaker Experiments)
- Depuración y perfilado (SageMaker Debugger)
- Gestión de Endpoints y Pipelines
- Integración nativa con Git

Analogía: Visual Studio Code / PyCharm para flujo de trabajo de ML, accesible desde navegador.

Alternativas: Vertex AI Workbench (GCP), Azure ML Studio (Azure).

SageMaker Data Wrangler

Herramienta de preparación de datos visual y de bajo código/sin código.

Funcionalidades:

- **Conectores de Datos**: Importar desde S3, Athena, Redshift, Snowflake, etc.
- **Análisis y Visualización**: Informes automáticos de calidad, histogramas

- **Transformaciones Integradas**: 300+ transformaciones preconfiguradas
  - Manejar valores atípicos
  - Codificar categóricas
  - Normalizar
- **Exportación de Código**: Genera código Python/PySpark automáticamente

Caso de Uso Ideal: Acelerar exploración y limpieza inicial de datos.

SageMaker Processing Jobs

Entorno gestionado para ejecutar scripts de procesamiento de datos a gran escala.

¿Por qué usarlo en lugar de notebook?

- **Escalabilidad**: Ejecuta script en clúster de instancias potentes
- **Desacoplamiento**: Notebook solo orquesta, trabajo se ejecuta independientemente
- **Reproducibilidad**: Entorno definido por contenedor Docker

Flujo:

1. Escribes script en Python con Scikit-learn o Spark
2. Lo subes a S3
3. Lanzas ProcessingJob especificando tipo de instancia, número, rutas S3

SageMaker Feature Store

Problema: Training-serving skew. Características usadas en entrenamiento son diferentes a las usadas en producción.

Solución: Repositorio centralizado para almacenar, recuperar, compartir características de ML.

Doble Almacenamiento:

1. **Online Store**: Para inferencia en tiempo real
   - Baja latencia
   - Optimizado para lecturas rápidas de una fila
2. **Offline Store**: Para entrenamiento
   - Bajo coste (S3)
   - Optimizado para escanear grandes volúmenes históricos

Beneficios: Consistencia entre entrenamiento e inferencia, reutilización entre equipos, gobernanza.

SageMaker Training Jobs: El Corazón del Entrenamiento

Similares a Processing Jobs, pero optimizados para entrenamiento de modelos. Abstracción completa de gestión de infraestructura.

API Estimator: En SDK Python de SageMaker, usas objeto Estimator para definir y lanzar trabajo.

Parámetros clave que defines:

- Contenedor Docker a usar (algoritmo integrado, framework, propio)
- Rol de IAM con permisos necesarios
- Tipo y número de instancias de cómputo
- Ubicación de datos de entrenamiento en S3
- Hiperparámetros

```
from sagemaker.xgboost import XGBoost

xgb_estimator = XGBoost(
    entry_point='train.py',
    role=role,
    instance_count=1,
    instance_type='ml.m5.xlarge',
    framework_version='1.5-1',
    hyperparameters={
        'max_depth': 5,
        'eta': 0.2,
        'objective': 'binary:logistic'
    }
)

xgb_estimator.fit({
    'train': s3_train_data,
    'validation': s3_val_data
})
```

Opción 1: Algoritmos Integrados (Built-in)

SageMaker ofrece colección de algoritmos de ML y DL populares, ya implementados y optimizados.

Ventajas:

- Alto rendimiento: Optimizados para grandes datasets y modo distribuido
- Fácil de usar: No escribes código de entrenamiento, solo configuras hiperparámetros

Ejemplos:

- **Clásicos**: Linear Learner, K-Means, PCA, Factorization Machines
- **Árboles**: XGBoost, LightGBM, Random Forest
- **Deep Learning**: Image Classification, Object Detection, Semantic Segmentation, BlazingText (NLP)

Opción 2: Soporte para Frameworks y BYOC

Contenedores de Frameworks (Deep Learning Containers)

- AWS proporciona y mantiene contenedores preconfigurados
- TensorFlow, PyTorch, Scikit-learn, MXNet
- Solo proporciona tu script de entrenamiento (train.py)
- SageMaker inyecta variables de entorno, datos, lo ejecuta

Bring Your Own Container (BYOC)

- Framework menos común u dependencias muy específicas
- Empaquetar tu entorno completo en contenedor Docker
- Subirlo a ECR (Elastic Container Registry)
- Decirle a SageMaker que lo use
- Máxima flexibilidad

Ahorrando Dinero: Managed Spot Training

¿Qué son Spot instances?

- Capacidad de cómputo no utilizada de EC2
- Vendida con descuento de hasta 90% sobre precio bajo demanda

El "pero": AWS puede reclamar la instancia con 2 minutos de preaviso.

SageMaker Managed Spot Training:

- Permite usar Spot instances para trabajos de entrenamiento (use_spot_instances=True)
- SageMaker gestiona interrupciones:
  - Guarda checkpoints periódicamente en S3
  - Si instancia se interrumpe, busca una nueva y reanuda desde último checkpoint
- **Impacto**: Puede reducir costes de entrenamiento de modelos grandes de forma espectacular

---

SESIÓN 9: Plataformas de Machine Learning Gestionadas II

El Ciclo de Vida del ML: ¿Dónde Estamos?

Tras explorar fases iniciales (comprensión del negocio, adquisición de datos, entrenamiento), ahora nos adentramos en la "última milla" del ML.

Despliegue: Llevar modelos entrenados a producción eficientemente
Monitorización: Vigilancia continua de rendimiento y deriva
Mantenimiento: Actualización y mejora a lo largo del tiempo

Objetivo final: Entregar valor real y sostenido a través de modelos que operan de forma fiable en el mundo real.

Artefactos del Modelo: El "Paquete" de Nuestro Trabajo

Tras un job de entrenamiento exitoso, obtenemos artefactos almacenados en bucket S3 como archivo comprimido: `model.tar.gz`

Este "paquete" es la unidad fundamental para el despliegue, siendo autocontenido y versionable.

Archivos del modelo:

- Pesos y arquitectura (.pth, .pb, .pkl)

Código de inferencia:

- `inference.py` con funciones: model_fn(), input_fn(), predict_fn(), output_fn()

Requerimientos:

- Dependencias en requirements.txt

Inferencia en Tiempo Real: SageMaker Real-time Endpoints

Concepto: Servicio HTTP/S persistente que aloja tu modelo para predicciones de baja latencia.

Ideal para:

- Chatbots interactivos
- Sistemas de recomendación en tiempo real
- Detección de fraude en transacciones

Características Clave:

- **Auto Scaling**: Ajusta automáticamente número de instancias según carga
- **Variantes de Modelos**: Desplegar múltiples modelos en mismo endpoint para A/B testing
- **Gateway**: API Gateway proporciona punto de entrada

Inferencia sin Servidor: SageMaker Serverless Inference

Concepto: Opción que abstrae completamente gestión de servidores. No hay instancias EC2 que configurar.

Caso de Uso Ideal: Cargas de trabajo intermitentes o esporádicas con largos periodos de inactividad.

Ventajas:

- Pagas solo por tiempo de cómputo y datos procesados
- Sin coste si no se usa
- Escalado automático de cero a lo necesario

Consideración: Posible "arranque en frío" (cold start) con mayor latencia inicial

Inferencia Asíncrona: Para Tareas Pesadas

Diseñada para predicciones que no requieren respuesta inmediata. Cliente envía petición, recibe confirmación de aceptación, procesamiento ocurre en segundo plano.

Caso de Uso Ideal:

- Payloads grandes (hasta 1GB): imágenes de alta resolución, documentos extensos
- Tiempos de procesamiento largos: modelos complejos que tardan minutos

Flujo de Trabajo:

1. Cliente envía petición al endpoint
2. Petición encolada en SQS gestionado por SageMaker
3. SageMaker procesa la petición de la cola
4. Resultado depositado en S3
5. Notificación opcional vía SNS

Transformación por Lotes: SageMaker Batch Transform

Concepto: Realizar inferencias sobre conjunto de datos completo de una sola vez. No requiere endpoint persistente.

Muy rentable: No hay coste de endpoint en standby.

Casos de Uso Ideales:

- Generación de informes diarios de segmentación
- Puntuación de leads de marketing al final del día
- Pre-procesamiento de grandes volúmenes

Flujo de Trabajo:

1. Inicias "trabajo de transformación"
2. SageMaker provisiona infraestructura necesaria
3. Lee datos de entrada desde S3
4. Ejecuta predicciones sobre todo dataset
5. Guarda resultados en otro bucket S3
6. Infraestructura se desprovisiona automáticamente

Despliegue en el Borde: SageMaker Edge Manager & Neo

No todas las inferencias ocurren en la nube. A veces se necesita IA en dispositivos con:

- Conectividad limitada
- Requisitos de latencia ultra-baja
- Procesamiento local por privacidad

SageMaker Neo

- Compilador de modelos que optimiza para hardware específico
- NVIDIA, Intel, ARM, etc.
- Genera ejecutables hasta 25x más rápidos y más pequeños sin pérdida de precisión

SageMaker Edge Manager

- Servicio para operar, monitorizar y actualizar flotas de modelos en dispositivos edge
- Empaqueta modelo optimizado con agente de software
- Gestiona despliegue seguro

La Realidad de la Producción: ¿Por Qué Monitorizar?

Un modelo desplegado no es el final del camino. El mundo real cambia constantemente.

1. Data Drift

- Distribución estadística de datos de entrada cambia
- Ejemplo: Modelo de recomendación de moda entrenado con datos de verano empieza a recibir datos de invierno

2. Concept Drift

- Relación entre variables de entrada y objetivo cambia
- Ejemplo: En modelo de fraude, estafadores inventan nuevas técnicas que el modelo nunca ha visto

3. Bias Drift

- Modelo empieza a hacer predicciones injustas o sesgadas para ciertos subgrupos demográficos
- Afecta a equidad del sistema

SageMaker Model Monitor

Función Principal: Detecta automáticamente desviación de calidad de un modelo en producción.

¿Cómo funciona?

1. **Línea Base (Baseline)**
    - Durante entrenamiento, se crea línea base de estadísticas y restricciones de datos
2. **Captura de Datos**
    - Endpoint captura porcentaje del tráfico de inferencia en tiempo real
3. **Comparación**
    - Periódicamente, trabajo compara estadísticas de datos capturados con línea base
4. **Alerta**
    - Si se detectan violaciones, genera informe y lanza alerta en CloudWatch

Explicabilidad y Sesgo: SageMaker Clarify

Función Principal: Proporciona visibilidad sobre comportamiento de modelos y posibles sesgos. Componente esencial para IA Responsible.

Análisis de Sesgo (Pre-entrenamiento)

- Mide sesgos en conjunto de datos inicial

Análisis de Sesgo (Post-entrenamiento)

- Evalúa si modelo produce predicciones sesgadas

Explicabilidad (Explainability)

- Transparencia y auditabilidad
- Ayuda entender por qué modelo tomó decisión específica
- Utiliza algoritmos como SHAP para calcular importancia de cada feature
- Genera informes visuales cruciales para reguladores y depuración

Equidad (Fairness) y Ética

- Se centra en salida de predicciones en relación con características protegidas
- Ejemplo: género, etnia, edad
- Métricas de Disparidad (ej. Tasa de Falsos Positivos Dispar)

MLOps: El DevOps para Machine Learning

MLOps es cultura y conjunto de prácticas que busca unificar desarrollo de modelos de ML (Dev) con su operación en producción (Ops).

Objetivos:

- Automatizar ciclo de vida completo del ML
- Aumentar velocidad de entrega de modelos
- Mejorar fiabilidad y reproducibilidad
- Facilitar colaboración entre equipos

En lugar de: Ejecutar notebooks manualmente
Crear: Flujos de trabajo automatizados, versionados, repetibles

---

SESIÓN 10: Servicios Cognitivos y de IA Aplicada

Introducción a los Servicios de IA Pre-entrenados

Concepto: Modelos de Machine Learning de alta complejidad, previamente entrenados por proveedores de nube sobre conjuntos de datos masivos, expuestos a través de una API.

Analogía: "Cerebros de IA bajo demanda". En lugar de construir y entrenar un cerebro desde cero, simplemente haces llamada a una API que ya sabe hacerlo.

Abstracción: Desarrollador no gestiona infraestructura, entrenamiento del modelo, ni su optimización. Solo se enfoca en consumo del servicio.

Ventajas Clave de las APIs de IA

1. **Aceleración del Desarrollo**

    - Integrar capacidades sofisticadas (reconocimiento facial, traducción) en días en lugar de meses/años

2. **Democratización de la IA**
    - No requiere equipo de científicos de datos
    - Desarrollador con conocimientos de APIs es suficiente
3. **Coste-Eficiencia**
    - Se evita inversión inicial en GPUs, recolección de datos, talento especializado
    - Modelo de pago pay-as-you-go
4. **Escalabilidad Gestionada**
    - Infraestructura escala automáticamente
    - Proveedor se encarga de actualizaciones y mantenimiento

Limitaciones y Consideraciones

1. **Menor Personalización**
    - Modelos entrenados para tareas genéricas
    - Si caso de uso es muy nicho, puede no ser suficiente
2. **Dependencia del Proveedor**
    - Vendor lock-in: Difícil migrar a otro proveedor
3. **Caja Negra**
    - A menudo sin control sobre arquitectura exacta del modelo, datos de entrenamiento, hiperparámetros
    - Problema para explicabilidad (XAI)
4. **Costes a Escala**
    - Aunque inicio es barato, volumen masivo de llamadas puede escalar significativamente

La Ética del "Pre-entrenado": Sesgos y Responsabilidad

El Problema del Sesgo (Bias)

- Modelos se entrenan con datos del mundo real
- Si datos contienen sesgos históricos (raciales, género, culturales), modelo los aprenderá

Ejemplo Famoso: Sistemas de reconocimiento facial con tasas de error más altas en mujeres de piel oscura vs. hombres de piel clara

Responsabilidad Compartida

- Proveedor nube: Responsable de mitigar sesgo en sus modelos
- Desarrollador: Responsable del impacto en aplicación y usuarios finales

Pregunta: ¿Quién es responsable si API de IA toma decisión sesgada que perjudica usuario?

Categorías de Servicios Cognitivos en AWS

Visión por Computador

- Análisis de imágenes y vídeos
- Detectar objetos, rostros, texto

Voz y Habla

- Conversión de texto a voz y viceversa
- Interacciones más naturales

Procesamiento del Lenguaje Natural (NLP)

- Análisis y comprensión del texto humano

Búsqueda y Descubrimiento

- Localización de información relevante
- Generación de recomendaciones

Servicios para Desarrolladores

- Herramientas de IA específicas para ciclo de desarrollo

Visión por Computador: Amazon Rekognition

¿Qué es?: Servicio que facilita adición de análisis de imágenes y vídeos a tus aplicaciones.

Capacidades Principales:

- **Detección de Objetos y Escenas**: Identifica miles de objetos (coche, árbol, perro) y escenas (playa, ciudad, interior)

- **Reconocimiento Facial**: Detecta rostros, analiza atributos (emociones, género, edad aproximada), compara caras para verificación de identidad
- **Análisis de Vídeo**: Detecta actividades (persona corriendo), rastrea personas en vídeo

Funcionalidades Avanzadas:

- **Moderación de Contenido**: Detecta contenido explícito o sugerente
- **Detección de Texto (OCR)**: Extrae texto de imágenes
- **Detección de EPP**: Identifica si personas llevan mascarillas, cascos, guantes (seguridad industrial)
- **Custom Labels**: Entrena Rekognition con pequeño set de imágenes propias

Voz y Habla: Polly & Transcribe

Amazon Polly

- Convierte texto en habla realista (TTS)
- **Voces Neuronales NTTS**: Calidad significativamente superior
- Amplia selección de voces en múltiples idiomas
- **SSML**: Lenguaje XML para controlar pronunciación, volumen, tono, velocidad

Amazon Transcribe

- Convierte audio en texto (STT)
- **Transcripción en Tiempo Real**: Para webinars, llamadas
- **Identificación de Hablantes**: Diferencia quién dijo qué
- **Vocabulario Personalizado**: "Enseña" términos específicos de tu dominio
- **Redacción Automática**: Puede ofuscar información sensible

Juntos: Base para asistentes de voz, sistemas de dictado, subtitulado automático.

NLP y Texto: Comprehend, Translate y Lex

Amazon Comprehend

- Extrae insights del texto mediante análisis de sentimientos, reconocimiento de entidades

- **Tareas Principales:**

    - Extracción de Entidades: Personas, lugares, fechas, organizaciones
    - Análisis de Sentimiento: Positivo, negativo, neutro, mixto
    - Detección de Idioma y Frases Clave
    - Modelado de Tópicos

- **Comprehend Medical**: Variante especializada con textos médicos

Amazon Translate

- Traduce texto entre idiomas

Amazon Lex

- Construye chatbots e interfaces conversacionales inteligentes

Criterios para Elegir un Servicio Cognitivo

1. **Precisión y Rendimiento**
    - ¿Modelo generalista es suficientemente bueno?
    - Hay benchmarks objetivos?
    - Siempre prueba con tus propios datos
2. **Opciones de Personalización**
    - ¿Necesitas entrenar con tus datos?
    - ¿Qué tan fácil es hacerlo?
3. **Soporte de Idiomas**
    - ¿Soporta idiomas y dialectos de tus usuarios?
4. **Facilidad de Integración**
    - ¿Ofrece SDKs para tu lenguaje?
    - ¿API está bien documentada?
5. **Modelo de Costes**
    - ¿Cómo se tarifica? (por llamada, por segundo de audio, por caracteres)
    - Estima costes a escala
6. **Cumplimiento y Soberanía**
    - ¿Dónde se procesan y almacenan datos?
    - ¿Cumple GDPR, HIPAA si necesario?

SESIÓN 12: Optimización de Costes y FinOps en Cloud para IA

FinOps - Más Allá de "Apagar lo que no usas"

FinOps (Cloud Financial Operations): Disciplina y práctica cultural que impulsa responsabilidad financiera y maximiza valor de negocio del gasto en la nube.

No se trata solo de ahorrar dinero: Se trata de ganar más dinero tomando decisiones de ingeniería informadas por su impacto financiero.

La Ecuación Clave:

```
Valor de Negocio = (Rendimiento del Modelo * Velocidad de Iteración) / Coste de Infraestructura
```

Analogía: Llevar agilidad de DevOps al ámbito financiero de la nube. En lugar de presupuesto fijo anual, gestionamos gasto variable y dinámico.

Los Principios Clave de FinOps

1. **Colaboración Interfuncional**
    - Equipos de Finanzas, Ingeniería (IA/ML), Negocio hablan el mismo idioma
    - Compartimos objetivos
    - Ingenieros entienden impacto de su código en la factura
2. **Propiedad y Responsabilidad**
    - Equipos que construyen y ejecutan servicios son responsables de su gasto
    - "Tú lo construyes, tú lo ejecutas, tú lo costeas"
3. **Decisiones Basadas en el Valor**
    - Cada euro gastado debe justificarse con retorno o valor para negocio
    - Ejemplo: ¿Entrenar este modelo una hora más mejorará precisión lo suficiente como para justificar coste de GPU?
4. **Visibilidad y Transparencia**
    - Equipos necesitan acceso fácil a sus datos de costes casi en tiempo real
5. **Optimización Continua**
    - Gestión de costes no es proyecto puntual
    - Proceso iterativo y constante

El Menú de Precios de la Nube

Modelo	Descripción	Descuento	Ideal para
On-Demand	Máxima flexibilidad, sin compromisos	Sin descuento	Cargas impredecibles
Reservas	Compromiso 1-3 años	Hasta 72%	Cargas estables
Spot	Capacidad excedente	Hasta 90%	Cargas tolerantes a fallos

Pago por Uso (On-Demand)

Concepto: Pagas por capacidad de cómputo por segundo/hora, sin compromisos a largo plazo.

Pros:

- Máxima flexibilidad
- Sin inversión inicial
- Lanza y termina cuando quieras

Contras:

- El más caro por hora

Ideal para:

- Desarrollo, pruebas, experimentos de IA (no conoces duración)
- Aplicaciones con picos de demanda a corto plazo, impredecibles
- Primera vez despliegas endpoint de inferencia para medir demanda real

¡La Trampa!: Es el modelo más caro. Dejar recursos On-Demand encendidos por inercia es principal fuente de gasto inesperado.

Instancias Reservadas y Savings Plans

Te comprometes a usar cierta cantidad de cómputo durante 1 o 3 años a cambio de gran descuento.

Standard RIs

- Mayor descuento
- Ligado a familia de instancia específica en región
- Menos flexible

Convertible RIs

- Menor descuento
- Permite cambiar familia de instancia, SO

EC2 Instance Savings Plans

- Descuento a cambio de compromiso de gasto por hora
- Más flexible que RIs

Compute Savings Plans

- MÁS flexible
- Aplica a cualquier cómputo (EC2, Fargate, Lambda) en cualquier región

Instancias Spot - El Secreto Mejor Guardado para IA

¿Qué son?: Capacidad de cómputo no utilizada vendida a precio muy bajo.

El Trato: Descuentos de hasta 90% sobre On-Demand.

La "Letra Pequeña": AWS puede reclamar instancia con 2 minutos de preaviso.

¿Por qué son perfectas para IA?

1. **Entrenamientos de ML**
    - Frameworks (TensorFlow, PyTorch) soportan checkpoints
    - Si instancia se interrumpe, reanuda desde último checkpoint en nueva instancia Spot
2. **Procesamiento por lotes**
    - ETL, preprocesamiento
3. **Simulaciones, renderizado**

Estrategias para Usar Spot con Éxito

1. **No te cases con una instancia**
    - Solicita múltiples tipos
    - Si un tipo no está disponible, clúster obtiene otro
2. **Diversifica entre Zonas de Disponibilidad (AZs)**
    - Disponibilidad de Spot varía por AZ
    - Lanza en múltiples AZs para aumentar probabilidad
3. **Utiliza servicios gestionados que lo simplifican**

- **AWS SageMaker Managed Spot Training**: Con simple parámetro (use_spot_instances=True), SageMaker gestiona checkpoints y reanudación. Puede ahorrar hasta 90%
- **AWS EC2 Fleet / Auto Scaling Groups**: Define flota con porcentaje de On-Demand (base) y Spot (escala)

¿De Dónde Vienen los Cargos? Principales Impulsores de Coste

Una solución de IA no es un servicio, es un ecosistema de recursos. Cada componente contribuye al coste.

1. Cómputo (compute) - "El motor"

- Las instancias (CPU/GPU) donde se entrenan modelos y se ejecutan inferencias
- Suele ser **70% del coste total**

2. Almacenamiento (storage) - "El combustible"

- Donde viven datasets, modelos entrenados, logs, backups

3. Transferencia de Datos (data transfer) - "Las tuberías"

- Mover datos dentro y fuera de la nube

4. Servicios Gestionados de IA/ML - "La inteligencia"

- Plataformas como SageMaker, APIs cognitivas

El Coste del Cómputo: El Gigante Silencioso

Una `p4d.24xlarge` de AWS puede costar **$30/hora On-Demand**.

¡Eso es $260,000 al año si se ejecuta 24/7!

Factores que influyen:

- Tipo de instancia: GPU vs. CPU, generación, tamaño
- Duración del trabajo: Entrenamiento 100 horas vs. 10 horas
- Escalado: Número de instancias para entrenamiento distribuido

Pregunta Clave: ¿Realmente necesito A100 para este experimento, o T4 sería suficiente?

Almacenamiento y Transferencia: Las Fugas Lentas

Almacenamiento

- Volumen de datos: Datasets Terabytes, modelos Gigabytes, logs acumulados
- Clase de almacenamiento: S3 Standard es 10x más caro que S3 Glacier Deep Archive
- Snapshots y Backups: Importantes, pero los olvidados pueden acumular costes

Transferencia de Datos

- **Salida de datos (Egress)**: ¡CUIDADO! Mover datos FUERA de la nube a internet tiene coste elevado
- Datos que entran (Ingress): Generalmente gratuitos
- Transferencia entre regiones: También tiene coste
- **NAT Gateways**: Muy útil pero puede generar costes altos si no se monitoriza

Servicios Gestionados de IA y Monitorización

AWS SageMaker (Ejemplo)

- **Notebooks/Instancias Studio**: Pagas por tipo y duración. ¡Apágalas cuando no las uses!
- **Training Jobs**: Pagas por tipo y duración de instancias de entrenamiento
- **Inference Endpoints**: Pagas por tipo y duración, independientemente de si reciben tráfico

APIs Cognitivas

- Coste basado en número de llamadas o cantidad de datos procesados

Logging y Monitorización

- Almacenamiento de logs y métricas personalizadas pueden sumar costes

Herramientas de Gestión de Costes

Nivel	Herramienta	Función
1: Visualización y Alerta	Cost Explorer, Budgets	Visión general y alertas tempranas
2: Análisis Profundo	Cost and Usage Report (CUR)	Análisis forenses y detallados
3: Recomendaciones Proactivas	Trusted Advisor, Cost Anomaly Detection	Optimización guiada

AWS Cost Explorer - Tu GPS Financiero

¿Qué es?: Interfaz visual para explorar costes y uso.

Funcionalidades:

- Visualización Histórica: Últimos 12 meses
- Filtrado y Agrupación: Por servicio, región, cuenta, etiquetas
- Previsiones Forecasting: Predice factura a final de mes
- Informes Guardados: Personalizados para tus proyectos de IA

Caso de Uso para IA: Agrupar costes por etiqueta `project:bert-training` para ver exactamente cuánto costó ese experimento.

AWS Budgets - Tu Perro Guardián

¿Qué es?: Establece umbrales de coste y recibe alertas cuando se superan.

Tipos de Presupuestos:

- **Coste**: Alerta cuando gasto total (o filtrado) supera $X
- **Uso**: Alerta cuando usas más de X horas de un tipo de instancia

¡Superpoder! - Budgets Actions
No solo alerta, sino que **puede actuar automáticamente**.

Ejemplo: Si presupuesto para entorno dev supera 100%, Budget Action aplica automáticamente política IAM que niega permiso para lanzar nuevas instancias de GPU, evitando catástrofe financiera.

CUR y Detección de Anomalías - Nivel Experto

Cost and Usage Report (CUR)

- Informe más detallado posible
- Fichero CSV o Parquet entregado a bucket S3
- Contiene cada cargo individual con granularidad horaria
- Para análisis muy avanzados usando Athena (SQL sobre S3) y QuickSight (BI)

AWS Cost Anomaly Detection

- Utiliza ML para aprender tu patrón de gasto normal
- Te alerta automáticamente si detecta gasto inusual
- Incluso si no ha superado tu umbral de presupuesto
- Ejemplo: Detecta que job de ETL empezó a escribir logs excesivos a S3

El Poder del Etiquetado (Tagging)

¿Qué es?: Metadatos (pares clave-valor) que asignas a tus recursos AWS.

¿Por qué es CRÍTICO?

- Sin etiquetas: Factura es bloque monolítico
- Con etiquetas: Desglosar costes por:
  - `project: face-recognition-v2`
  - `environment: development, staging, production`
  - `team: data-science-alpha`
  - `owner: profesor-x`

Estrategia de Tagging: Consistencia es clave. Define política obligatoria y úsala en toda la organización.

SESIÓN 13: Infraestructura como Código (IaC)

¿Qué es la Infraestructura como Código (IaC)?

Definición: Proceso de gestionar y aprovisionar centros de datos e infraestructura a través de ficheros de definición legibles por máquina, en lugar de usar configuración manual o herramientas de interfaz gráfica.

Analogía: El plano de construcción de un arquitecto para tu infraestructura en la nube. En lugar de que los obreros (administradores de sistemas) decidan dónde poner cada ladrillo (recurso) sobre la marcha, siguen un plan detallado y versionado.

El Mundo ANTES de IaC: El "ClickOps"

El Problema: Gestión manual a través de consolas web o scripts aislados.

Consecuencias Directas:

1. **"Snowflake Servers"**: Servidores únicos, configurados manualmente, imposibles de replicar
2. **Deriva de Configuración**: Entornos dev, test, prod que divergen con el tiempo
3. **Falta de Trazabilidad**: ¿Quién cambió qué y por qué? Difícil auditar
4. **Proceso Lento y Propenso a Errores**: Aprovisionamiento manual es lento y sujeto a error humano

Principios Clave de IaC

1. Repetibilidad

- Capacidad de crear el mismo entorno exacto cada vez que se ejecuta el código
- Crucial para experimentación en ML donde necesitas clonar stack completo para nuevo modelo

2. Idempotencia

- Aplicar misma configuración múltiples veces produce mismo resultado final
- Ejemplo: Si código dice "debe existir bucket S3 llamado mi-dataset-proyecto-x", primera vez lo crea. Siguientes veces, detecta que existe y no hace nada

3. Versionado

- Los ficheros IaC son código, se gestionan con Git
- Historial completo de cambios
- Capacidad de revertir (rollback)
- Colaboración en equipo mediante pull requests

4. Automatización

- Despliegue y gestión de infraestructura se convierten en paso más de pipeline automático
- Elimina intervención manual

5. Documentación Viva

- El propio código se convierte en fuente de verdad y documentación más actualizada
- No más diagramas de Visio desactualizados

Ventajas de IaC para Infraestructuras de IA

Los proyectos de IA no son solo código de Python; dependen masivamente de infraestructura compleja y a menudo efímera. IaC es el habilitador clave de MLOps.

1. **Despliegue Rápido**
    - ¿Nuevo Data Scientist se une al equipo? ¿Necesitas probar nueva arquitectura de red neuronal?
    - Despliega entorno completo con GPUs, almacenamiento, red en minutos
2. **Consistencia entre Entornos**
    - Garantiza que entorno donde entrenas modelo (dev/training) es idéntico al de inferencia (prod)
    - Elimina sorpresas
3. **Reproducibilidad**
    - Para que resultado científico o modelo de negocio sea válido, debe ser reproducible
    - IaC garantiza que infraestructura subyacente es parte de esa reproducibilidad

Enfoques: Declarativo vs. Imperativo

Declarativo ("Qué")

- Defines el estado final deseado
- Le dices a herramienta: "Quiero una VM t3.large con este disco y en esta red"
- La herramienta se encarga de la lógica para alcanzar ese estado
- Ejemplos: Terraform, CloudFormation, ARM Templates, Kubernetes YAML
- **Enfoque dominante y preferido para IaC**

Imperativo ("Cómo")

- Defines la secuencia de comandos a ejecutar
- Le dices: "Primero, crea clave SSH. Luego, ejecuta comando para crear VM..."
- Tú eres responsable de la lógica, manejo de errores, estados intermedios
- Ejemplos: Scripts usando AWS CLI, Azure CLI, SDKs de Python (Boto3)

Herramienta #1: AWS CloudFormation

¿Qué es?: El servicio de IaC nativo y totalmente gestionado de AWS. Es el lenguaje fundamental con el que AWS define sus propios servicios.

Componentes Clave:

- **Templates (Plantillas)**: Ficheros de texto en JSON o YAML que describen pila de recursos de AWS
- **Stacks (Pilas)**: Conjunto de recursos que se crean y gestionan como una única unidad a partir de un template
- **Change Sets (Conjuntos de Cambios)**: Permite previsualizar cambios que CloudFormation realizará antes de aplicarlos

Anatomía de un Template:

```
AWSTemplateFormatVersion: '2010-09-09'
Description: A simple S3 bucket for AI datasets
Parameters:
  ProjectName:
    Type: String
    Description: Name of the project to prefix the bucket name
Resources:
  AIDataBucket:
    Type: 'AWS::S3::Bucket'
    Properties:
      BucketName: !Sub '${ProjectName}-ai-datasets-${AWS::AccountId}'
```

```
      VersioningConfiguration:
        Status: Enabled
      Tags:
        - Key: "Project"
          Value: !Ref ProjectName
Outputs:
  BucketName:
    Description: "Name of the S3 bucket created"
    Value: !Ref AIDataBucket
```

Herramienta #2: Terraform (by HashiCorp)

¿Qué es?: Herramienta de IaC open-source y agnóstica a la nube (multi-cloud). Ha se convertido en estándar de facto en la industria.

Diferencias Clave con CloudFormation:

- **Multi-Cloud**: Un solo lenguaje para gobernar AWS, Azure, GCP
- **Lenguaje HCL**: HashiCorp Configuration Language, diseñado para ser más legible que JSON/YAML
- **Gestión de Estado**: Terraform mantiene fichero de estado (.tfstate) que mapea recursos del mundo real con tu configuración

El Flujo de Trabajo:

1. **terraform init**
    - Se ejecuta una vez por proyecto
    - Descarga los providers necesarios
    - Prepara backend para fichero de estado
2. **terraform plan**
    - Equivalente al "Change Set" de CloudFormation
    - Lee estado actual, lo compara con código
    - Muestra plan de ejecución detallado
3. **terraform apply**
    - Aplica los cambios descritos en el plan
    - Pide confirmación explícita

Mejores Prácticas de IaC

1. **Versiona TODO en Git**
    - Sin excepciones
    - Todo cambio debe estar documentado y versionado
2. **Realiza cambios pequeños e incrementales**
    - Facilita revisión y rollback en caso de problemas
3. **¡Prueba tu código!**
    - Herramientas de linting y validación
    - Despliega en entornos de prueba primero
4. **Modulariza tu código**
    - Crea módulos reutilizables para componentes comunes
    - Ejemplo: Una VPC, una instancia EC2 segura
5. **NUNCA incrustes secretos**
    - No pongas contraseñas, claves de API, etc.
    - Usa AWS Secrets Manager o Parameter Store
6. **Gestiona el estado de Terraform de forma remota y segura**
    - Backend de S3 con bloqueo de estado (DynamoDB)
    - Previene corrupción del estado en equipos

SESIÓN 14: Casos de Estudio y Tendencias Futuras

Caso 1: Sistema de Recomendación Personalizado

Objetivo: Ofrecer recomendaciones de productos o contenidos en tiempo real a millones de usuarios.

Componentes Clave de Arquitectura:

1. **Ingesta de Datos**: Captura de interacciones de usuario (clics, vistas, compras)
2. **Almacenamiento**: Persistencia de perfiles de usuario y catálogo de ítems
3. **Entrenamiento del Modelo**: Procesamiento de datos y entrenamiento (filtrado colaborativo, basado en contenido, híbrido)
4. **Inferencia en Tiempo Real**: API de baja latencia que devuelve recomendaciones
5. **Ciclo de Feedback**: Reincorporación de nuevas interacciones para reentrenar el modelo

Arquitectura de Referencia en AWS:

- **Ingesta**: Amazon Kinesis Data Streams/Firehose
- **Almacenamiento**: S3 Data Lake para datos crudos/procesados, DynamoDB para perfiles de usuario y catálogo
- **Entrenamiento**: Amazon SageMaker usando algoritmos integrados (Factorization Machines)
- **Inferencia**: SageMaker Endpoints como API autoescalable
- **Solución Gestionada**: Amazon Personalize abstrae gran parte

Caso 2: Análisis de Imágenes Médicas

Objetivo: Crear sistema que analice imágenes médicas para ayudar a radiólogos a detectar anomalías.

Consideraciones Críticas:

- Seguridad, privacidad y cumplimiento normativo (HIPAA)

Componentes:

1. **Ingesta Segura**: Transferencia encriptada de archivos (formato DICOM)
2. **Entrenamiento de Modelo**: CNNs en hardware especializado (GPUs)
3. **Almacenamiento Conforme**: Control de acceso estricto, encriptación, auditoría
4. **Inferencia y Revisión Humana**: API para predicción + interfaz para validación por expertos
5. **Preprocesamiento**: Normalización de imágenes, aumento de datos

Arquitectura en AWS:

- **Ingesta/Transferencia**: AWS DataSync o Storage Gateway
- **Almacenamiento**: S3 con encriptación (SSE-S3/KMS), políticas IAM estrictas
- **Entrenamiento**: SageMaker con instancias GPU (P3, G4)
- **Inferencia**: SageMaker Endpoints
- **Revisión Humana**: Amazon A2I (Augmented AI)
- **Auditoría**: CloudTrail y CloudWatch

Caso 3: Análisis de Sentimiento en Redes Sociales

Objetivo: Monitorizar menciones de una marca y clasificarlas como positivas, negativas o neutras en tiempo real.

Componentes:

1. **Ingesta de Datos**: Conexión a APIs de redes sociales
2. **Procesamiento/Análisis**: Limpieza del texto y clasificación del sentimiento
3. **Almacenamiento**: Datos de texto crudos
4. **Visualización**: Dashboards para monitorizar resultados

Decisión Clave: ¿Usar API pre-entrenada o entrenar modelo personalizado?

Arquitectura en AWS:

- **Ingesta**: Lambda functions + Kinesis Firehose
- **Almacenamiento**: Kinesis Firehose → S3
- **Procesamiento:**
  - Opción 1: Lambda que llama a Amazon Comprehend (API de NLP)
  - Opción 2: SageMaker (BlazingText o modelo de Hugging Face)
- **Visualización**: Amazon OpenSearch Service o QuickSight

Caso 4: Detección de Fraude en Transacciones Financieras

Objetivo: Analizar transacciones de tarjeta de crédito en tiempo real para bloquear fraudulentas.

Consideraciones Críticas:

- Latencia ultra baja (decenas de milisegundos)
- Alta disponibilidad y reentrenamiento continuo

Componentes:

1. **Ingesta de Transacciones**: Stream de eventos de alta velocidad
2. **Ingeniería de Características**: Enriquecimiento en tiempo real (ej. ¿cuántas compras en última hora?)
3. **Motor de Inferencia**: Modelo de ML optimizado para latencia mínima
4. **Sistema de Alertas**: Notificación inmediata
5. **Reentrenamiento Continuo**: Modelo se adapta a nuevos patrones

---

Preguntas Frecuentes para Examen

Preguntas de Razonamiento y Justificación

1. Impacto de la Virtualización

Explica por qué la virtualización es el pilar tecnológico fundamental que hizo posible el modelo técnico del Cloud Computing.

Respuesta: La virtualización permite abstraer el hardware del software. Múltiples máquinas virtuales pueden ejecutarse en un único servidor físico, compartiendo hardware pero manteniéndose lógicamente aisladas. Esto permite a los proveedores de nube:

- Maximizar la utilización del hardware físico
- Proporcionar a los clientes "máquinas" bajo demanda
- Ofrecer elasticidad y flexibilidad
- Aislar clientes entre sí (multi-tenancy)

2. CapEx vs. OpEx

Describe la diferencia y justifica por qué el Cloud Computing cambia la balanza.

Respuesta:

- **CapEx**: Gasto de capital (compra de servidores). Gran inversión inicial, riesgo de sobreaprovisionar
- **OpEx**: Gasto operativo (servicios cloud). Predecible y variable, pagas solo por lo que usas
- Cloud cambia esto porque democratiza el acceso a infraestructura - una startup puede usar la misma tecnología que Google sin inversión inicial masiva

3. Seguridad en la Nube

Explica el Modelo de Responsabilidad Compartida con un ejemplo.

Respuesta: El proveedor es responsable de la seguridad DEL cloud (infraestructura física, hipervisor), mientras el cliente es responsable de la seguridad EN la cloud (sus datos, SO, configuración de firewalls).

- Ejemplo: AWS asegura que el disco S3 no falle. TÚ aseguras que está encriptado y tienes políticas de IAM correctas.

4. Arquitectura de Instancias

Un proyecto de IA de 3 capas (web, API de inferencia, base de datos). ¿Qué instancias usarías para cada una?

Respuesta:

- **Web**: Serie T (t3.large) - tráfico bajo con picos, rendimiento ampliable
- **API Inferencia**: Serie Inf o G - optimizadas para baja latencia de inferencia
- **Base de Datos**: Familia R o serverless RDS - optimizadas para memoria

Preguntas de Casos de Estudio

Caso: API de Análisis de Sentimiento

Startup con API exitosa pero tráfico impredecible, presupuesto limitado, punto de entrada IP fijo.

Preguntas:

1. ¿Qué servicio para IP fija? **Respuesta**: Elastic IP
2. ¿Qué dos servicios para escalabilidad automática? **Respuesta**: Auto Scaling Group + Elastic Load Balancer
3. ¿Qué modelo de precios para costes mínimos? **Respuesta**: Spot Instances (con On-Demand como base para disponibilidad)

FIN DE LA GUÍA COMPLETA DE ESTUDIO

Este documento contiene la totalidad del contenido de las 14 sesiones del curso de Cloud Computing para Inteligencia Artificial. Está organizado tema a tema, con explicaciones detalladas, ejemplos prácticos, conceptos clave y preguntas de repaso para prepararse para el examen.