# Actividad 3. Computación en la Nube

## A. Cuestionario

1. **Motivaciones para la nube**

    - (a) ¿Qué problemas o limitaciones existían antes del surgimiento de la computación en la nube y cómo los solucionó la centralización de servidores en data centers?

      Antes de la computación en la nube cada usuario/empresa que requiriera de una infraestructura de software debía comprar enteramente el hardware. Esto involucra el diseño, el mantenimiento y el costo de esta tarea. Es decir, estaba sujeto a precios elevados de hardware y el costo de un equipo encargado del correcto funcionamiento y manutención, así como, si dado el caso de aumentar las necesidades de potencia, se debía comprar nuevamente hardware costoso.

      La centralización en data centers se encargan de realizar estas tareas, a través de poner a disposición de sus clientes arquitecturas de hardware, a través de un costo (menor al de la compra y manutención), que además se puede escalar de manera eficiente y directa.

    - (b) ¿Por qué se habla de “The Power Wall” y cómo influyó la aparición de procesadores multi-core en la evolución hacia la nube?

      Se hace referencia como "the power wall" a la limitación de la frecuencia de CPU, diseño de CPU, y rendimiento de CPU causados por las restricciones térmicas o eléctricas que causarían inestabilidad en los procesadores.

2. **Clusters y load balancing**

    - (a) Explica cómo la necesidad de atender grandes volúmenes de tráfico en sitios web condujo a la adopción de clústeres y balanceadores de carga.

      La necesidad de atender grandes volúmenes de tráfico en sitios web conlleva a la adopción de clústeres debido a que sumar más servidores con una misma capacidad y dividirse la carga es más costo-eficiente que aumentar la potencia del hardware utilizado. A esta solución se le conoce como adición de nodos al cluster, en la que cada nodo está ejecutando la aplicación. Así, también surge la técnica de load balancing, la cual permite evitar la sobrecarga de un servidor, particionándolo entre otros nodos.

    - (b) Describe un ejemplo práctico de cómo un desarrollador de software puede beneficiarse del uso de load balancers para una aplicación web.

Un desarrollador de software, encargado de mantener el servicio de matrícula de una universidad, conoce que en la época de matrículas se produce un alto tráfico de usuarios y con muchas peticiones. Esto puede, y llega, a sobrecargar muchos sistemas. Así, tras el uso de un load balancer, se previene la caída completa de la página de matrícula, pues se mantiene varias instancias de la aplicación.

3. **Elastic computing**

    - (a) Define con tus propias palabras el concepto de Elastic Computing.

      Es el dinamismo en el uso de hardware necesario para ejecutar una aplicación en la nube variante según la demanda presente en la aplicación.

    - (b) ¿Por qué la virtualización es una pieza clave para la elasticidad en la nube?

      Para el uso de elastic computing, debemos notar que en el mismo hardware se ejecutan más de una aplicación. La virtualización permite la contenerización de las aplicaciones, facilitando el proceso de ejecución, pues si ocurren bugs dentro de una aplicación, esta no afecta a otras https://www.elastic.co/blog/how-to-handle-elasticsearch-virtualization

    - (c) Menciona un escenario donde, desde la perspectiva de desarrollo, sería muy difícil escalar la infraestructura sin un entorno elástico.

      Hemos de notar que el elastic computing nos permite no tener todo el hardware siendo utilizado al mismo tiempo. Así, si, por ejemplo, Amazon.com, deseara diseñar su plataforma sin elastic computing, teniendo en cuenta el alto tráfico que se genera en distintas épocas del año, sería complicado el diseño tomando en cuenta zonas, fechas y minimizando el costo. Esto resultaría en o bien una aplicación extremadamente cara de mantener, o una aplicación de poca confiabilidad en su estabilidad, pues caería ante altas fluctuaciones en el tráfico.

4. **Modelos de servicio (IaaS, PaaS, SaaS, DaaS)**

    - (a) Diferencia cada uno de estos modelos. ¿En qué casos un desarrollador optaría por PaaS en lugar de IaaS?

      Por ejemplo, para una aplicación la cual requiera de una base de datos sencilla sin la necesidad de especificación de versión de sistema operativo o control del versionado de kernel.

    - (b) Enumera tres ejemplos concretos de proveedores o herramientas que correspondan a cada tipo de servicio.

IaaS: Azure VM, Google Compute Engine, Google Cloud Storage

PaaS: Supabase, Firebase, Google CloudRun

SaaS: Spotify, Office365, Salesforce

DaaS: MongoDB Atlas, Dizzion, Citrix DaaS

5. **Tipos de nubes (Pública, Privada, Híbrida, Multi-Cloud)**
    - (a) ¿Cuáles son las ventajas de implementar una nube privada para una organización grande?

      Las ventajas de implementar una nube privada son la privacidad y seguridad de sus datos, control y flexibilidad, así como la libertad para hacer cambios a nivel arquitectura, elección de hardware y software a elección.

    - (b) ¿Por qué una empresa podría verse afectada por el “provider lock-in”?

      Pues al disponer de un servicio en la nube y querer realizar una migración, se presentan problemas, pues puede que se haga uso de servicios exclusivos del proveedor original de nube, así como un equipo de desarrollo con poca experiencia en la nube destino, lo cual incrementará costos y requerirá de una inversión mayor de tiempo para cumplir con sus propósitos reales.

    - (c) ¿Qué rol juegan los “hyperscalers” en el ecosistema de la nube?

      Son grandes proveedores de servicios de nube. Estos permiten la ejecución óptima de una aplicación en diversas partes del mundo minimizando latencia, correcto desempeño, y mejorando la experiencia del usuario con agilidad.

# B. Actividades de investigación

## E studio de casos

### Coca-Cola

**Motivaciones**

Tras 20 años operando en sus propias instalaciones, Coca-Cola experimentó una sobrecarga en sus servidores debido a un pico de tráfico de su sitio web durante un comercial del Super Bowl.

Coca Cola Super Bowl 2013 TV Spot: The Chase

Como se explica en este artículo de AWS, este fue el incidente que impulsó la decisión de migrar a la nube de AWS.

**Beneficios**

Tras la migración vieron una reducción de costos operativos del 40 %, y la eficiencia operativa también aumentó porque se redujo en un 80 % el volumen de tickets de TI. Además, la empresa aceleró su innovación, con lo cual desarrollo y desplegó nuevas soluciones como la experiencia de usuario sin contacto para su plataforma de dispensación de bebidas en solo 5 meses.

**Desafíos**

Es común que los desafíos no se compartan públicamente, así que en este caso solo podemos suponer cuáles fueron. Lo más probable es que hayan enfrentado retos relacionados con la seguridad de los datos, el cumplimiento normativo y la gestión de cambio organizacional.

### Nestlé USA

**Motivaciones**

Almacenar y gestionar e integrar todos los datos repartidos en los múltiples sistemas locales de esta compañía era muy difícil. Como se explica en este artículo, la necesidad de una visión unificada y la capacidad de implementar análisis avanzados motivaron la decisión de migrar a la nube de Azure.

**Beneficios**

Se eliminaron los silos y se logró que los datos sean reutilizables para todas las funciones empresariales, lo cual mejoró la toma de decisiones al poder hacer uso de motores de recomendación de ventas o IA. Además, siguiendo la metodología DevOps, se implementaron canalizaciones CI/CD, lo cual aumentó la eficiencia operativa al disminuir el tiempo de desarrollo y también el tiempo necesario para la ingesta de grandes conjuntos de datos

**Desafíos**

Se integraron datos de más de 10 fuentes y se desactivaron 17 sistemas legacy aislados, lo cual requirió una planificación muy cuidadosa.

## Comparativa de modelos de servicio

| Aspecto | Iaas | PaaS |
|---|---|---|

| | | |
|---|---|---|
| Gestión del hardware | Proveedor | Proveedor |
| Red y almacenamiento | Proveedor | Proveedor |
| Seguridad de infraestructura | Proveedor | Proveedor |
| Parches del SO | Operaciones | Proveedor |
| Escalado automático | Operaciones | Proveedor |
| Instalación del SO | Desarrollo / Operaciones | Proveedor |
| Middleware | Desarrollo / Operaciones | Proveedor |
| Base de datos | Desarrollo / Operaciones | Proveedor |
| Configuración del entorno | Desarrollo / Operaciones | Proveedor y Desarrollo |
| Seguridad de app | Desarrollo / Operaciones | Proveedor y Desarrollo |
| Actualizaciones de software | Desarrollo / Operaciones | Proveedor y Desarrollo |
| Despliegue de apps | Desarrollo | Desarrollo |

En un modelo SaaS, la empresa usuario no necesita ni equipo de desarrollo ni de operaciones para usar la aplicación porque el código ya está desarrollado y no hay infraestructura que administrar por parte de esta empresa. Lo único que hace la empresa es consumir el servicio y, a lo mucho, administrar opciones de usuario, seguridad básica e integraciones sencillas, como en Gmail o Microsoft 365.

## Estrategia de migración multinube

Nuestros objetivos para esta estrategia son los de minimizar el riesgo de dependencia de un solo proveedor y así mejorar la resiliencia y continuidad del negocio.

Se van a mantener:

- Sistemas críticos del negocio
- Almacenamiento principal de datos

Se van a migrar:

- Servicios web no críticos, pero con alta demanda
- Copias secundarias de bases de datos

Vamos a implementar una réplica secundaria en el segundo proveedor en caso de contingencia (failover automático). También vamos a usar herramientas como AWS Database Migration Service para la migración y Azure Data Sync para la sincronización de las nubes.

También necesitamos políticas de cumplimiento normativo homogéneas en ambas nubes. Esto se puede conseguir con Azure ADD y GDPR o HIPAA.

## Debate sobre costos

| Tipo de nube | Pros | Contras |
|---|---|---|
| Nube pública | Escalable, sin inversión inicial, ideal para startups. | Menos control, posibles problemas de cumplimiento, cambio de proveedor complejo. |
| Nube privada | Control total, alto cumplimiento, seguridad personalizada. | Costosa, poco flexible ante picos de demanda. |
| Nube híbrida | Balance entre control y escalabilidad, adaptable. | Complejidad en la gestión e integración. |
| Multinube | Redundancia, evita la dependencia de un solo proveedor (vendor lock-in), flexibilidad estratégica | Mayor complejidad técnica, costos operativos elevados. |

# C. Ejercicio de presentación de "mini-proyecto"

## Objetivos del sistema

El mini-proyecto consistirá en una plataforma web para la reserva de boletos de cine, la cual permitirá a los usuarios consultar la cartelera, seleccionar una función, escoger sus asientos y realizar los pagos en línea.

La aplicación también incluirá módulos para la gestión de salas, películas, horarios, pagos y notificaciones al usuario.

Por lo tanto, la aplicación estará orientada tanto para usuarios finales como para los cines.

## Selección de modelo de servicio: PaaS (Platform as a Service)

Elegimos el modelo PaaS porque permite al equipo de desarrollo centrarse en escribir código y desplegar funcionalidades sin preocuparse por la gestión de la infraestructura (servidores, redes, balanceadores de carga)

PaaS nos proporciona escalabilidad automática, gestión de bases de datos, herramientas de CI/CD y seguridad por defecto.

Algunas razones específicas por las que PaaS es ideal para este proyecto son:

- **Despliegue ágil** de nuevas versiones de la aplicación
- **Escalado automático** cuando haya mucho tráfico (fines de semana, estrenos importantes, días de promociones)
- Fácil integración con servicios externos (APIS de pagos, sistemas de mensajería)
- Reducimos costos operativos al no tener que mantener servidores

## Tipo de nube: Pública

Ventajas:

- Costo inicial bajo
- Alta disponibilidad y redundancia
- Acceso a bases de datos, balanceadores y monitoreo
- Escalabilidad bajo demanda

Desventajas:

- Dependencia del proveedor
- Seguridad delegada

# Esquema de escalabilidad

Arquitectura y tecnologías seleccionadas:

- Frontend (Vue.js)
    - El código estático del frontend se compilará y desplegará en AWS Amplify Hosting o (Amazon S3 + CloudFront)
    - Con esto se podrá entregar los assets desde un CDN global, asegurando la baja latencia en cualquier parte del mundo.
- Backend (REST o GraphQL)
    - Se desarrollará en Node.js
    - Se desplegará en AWS Elastic Beanstalk
    - Este último servicio es el que maneja automáticamente el balanceo de carga, la escalabilidad automática y el monitoreo.
- Bases de datos (PostgreSQL)
    - Se utilizará Amazon RDS for PostgresSQL
    - RDS nos facilita los backups automáticos y réplicas para lectura (según el tráfico)
    - Esto nos garantiza la **alta disponibilidad**
- Caché y colas
    - Amazon ElastiCache (Redis) para manejar el caché de las consultas frecuentes (horarios de películas y asientos disponibles)
    - Amazon SQS para eventos de forma asincrónica (confirmación de pagos, envío de tickets por correo)
- Notificación de eventos
    - Amazon SES (para los emails) y Amazon SNS (para las notificaciones push o por SMS)

# Costos y riesgos

Costos principales según los servicios:

| Componente | Servicio AWS | Costo estimado (mensual base) |
|---|---|---|
| Frontend (Vue) | S3 + CloudFront | $5–20 USD según tráfico |
| Backend | Elastic Beanstalk/App Runner | $20–60 USD por instancia pequeña |
| Base de datos | RDS PostgreSQL (db.t3.medium) | $50–100 USD con backups |
| Caché | ElastiCache (Redis) | $15–30 USD |
| Notificaciones | SES, SNS | Depende del volumen (ej. $0.10/1K mails) |
| Balanceador de carga | ELB (si usa Beanstalk) | $18–25 USD |
| Tráfico saliente (data out) | AWS Transfer Data Out | $0.09/GB después del primer GB |

Riesgos asociados:

- Vendor lock-in (dependencia del proveedor)
- Mal diseño de escalado, lo que se traduce a facturas altas
- Fallos del servicio del proveedor
- Requisitos legales o regulatorios: Debido a la información sensible (como los pagos)

# Presentación final