INTRODUCCIÓN AL CLOUD

Influencia del Cloud en la asignatura

La carga de trabajo puede variar

Elasticidad → el Cloud aporta capacidad rápidamente

Las máquinas pueden fallar

Disponibilidad → aumenta garantía de funcionamiento

Workload

Workload se refiere al tipo de trabajo que se realiza el computador:

- Servicios web
- Bases de datos
- Batch jobs (procesamiento por lotes)
- Workflows
- Machine learning
- Big Data

……

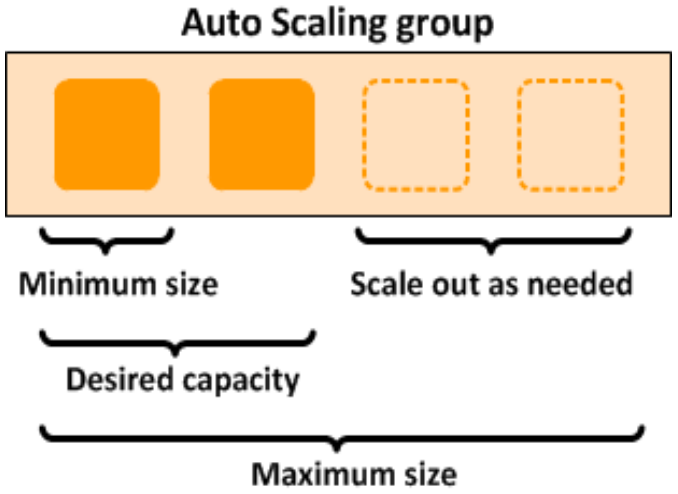
… y como su demanda cambia a lo largo del tiempo.

QoS (de capacidad y de disponibilidad)

Incremento de capacidad

Elasticidad: capacidad de adaptarse a la demanda con rapidez

El cloud ofrece el servicio de auto-escalado (Auto Scaling), configurable mediante “templates”



- Servicio gratuito
- Todas las VM usan la misma (AMI) Amazon Machine Image
- Mezcla de modelos de precios
- Mezcla de tipos de instancias
- Escalado dinámico o planificado

Incremento de disponibilidad

El cloud permite contratar soluciones de alta disponibilidad

- Uso de “Availability Zones”
- Despliegue en pares de regiones
- Almacenamiento con opciones de replicación de datos
- Posibilidad de “migrar” máquinas virtuales
- El auto-escalado controla el correcto funcionamiento de las máquinas (capacidad deseada)

Un paso mas… "Contenedores"

Definición

Es un software que encapsula una aplicación (funcionalidad), junto con todas sus dependencias (bibliotecas, etc)

El contenedor usa los recursos básicos del sistema operativo base

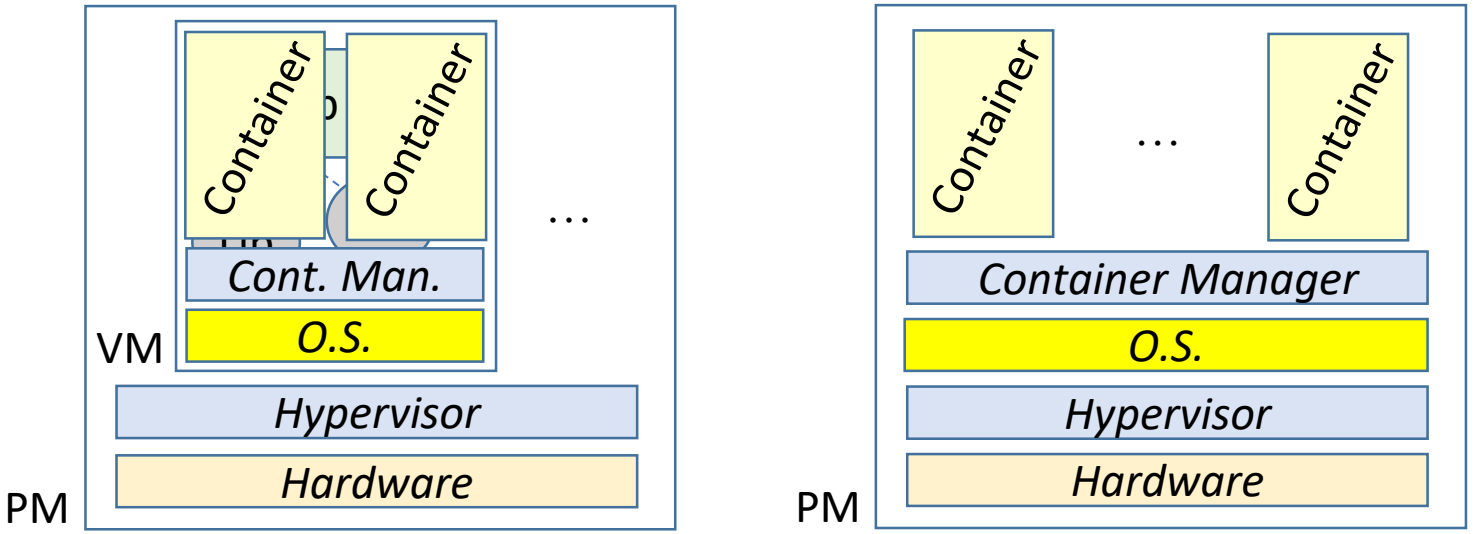
El software contenido en el contenedor se denomina imagen

La imagen se ejecuta sobre un gestor de contenedores

Funcionamiento

El contenedor constituye una funcionalidad aislada

Disponible sobre máquina física, virtual o *CaaS*



Google Container Engine (GKE), Amazon EC2 Container Service (ECS) y Microsoft Azure Container Service (ACS)

Beneficios de los contenedores

Los contenedores tienen ventajas frente a las máquinas virtuales:

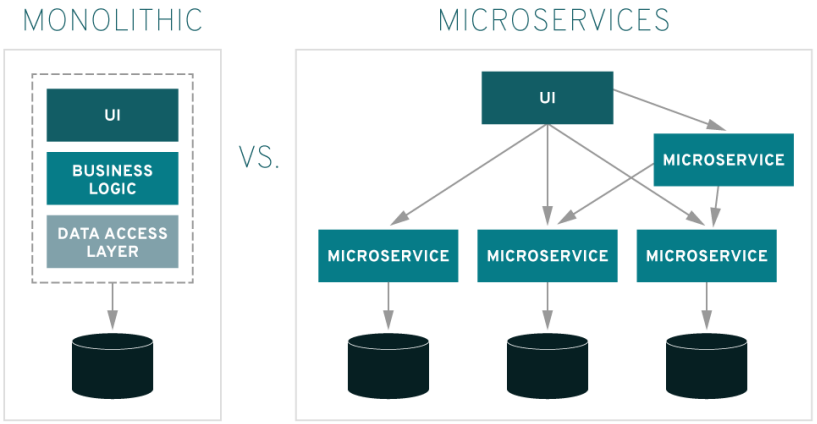
- Los contenedores son más ligeros que las maquinas virtuales (trabajan directamente sobre el Kernel)
- No es necesario instalar un sistema operativo por contenedor
- Arranca mucho más rápido que la máquina virtual
- Mejor portabilidad
- Asignación a nivel de grado fino de recursos “milicores”
- Posibilidad de desplegar una mayor cantidad de contenedores

Algo más … Microservicios

Las aplicaciones de la nube “cloud native applications” se basan:

En pequeños trozos de código con funcionalidad limitada → *microservicio*

Las aplicaciones se construyen combinando *microservicios*



INTRODUCCIÓN AL CLOUD

Ventajas

- Facilidad de desarrollo
- Facilidad de implementación (lenguaje más apropiado)
- Facilidad de despliegue (en entornos distribuidos)
- Facilidad de escalado

Desafíos

- Aplicaciones más complejas
- Importancia del grafo de dependencias

Microservicios y Contenedores

Los microservicios se despliegan en uno o varios contenedores

Para gestionar los contendores → Herramienta de *orquestación*

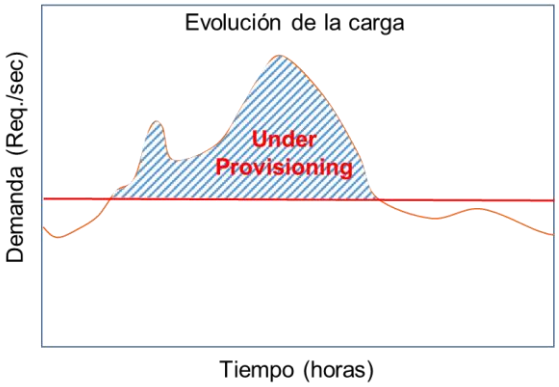
- Kubernetes
- Docker Swarm

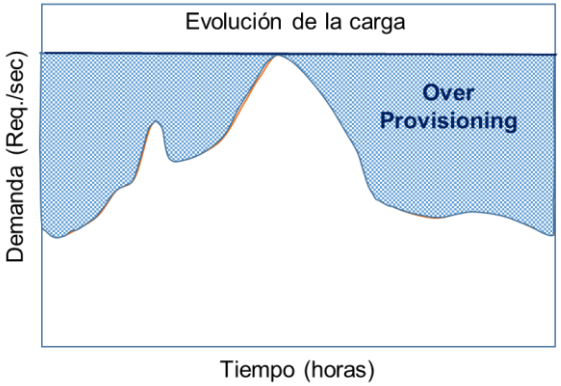
Aparece el concepto de "*pod*" → unidad mínima de planificación

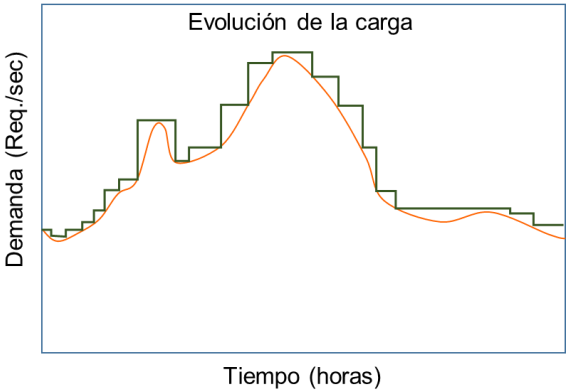
- *pod* → uno/varios contenedores que comparten almacenamiento y red
- Por simplicidad: *pod* → contenedor
- Aparece el nodo de gestión y el nodo de trabajo (*worker*)

Contenedores y Auto-escalado

Como se adaptan los recursos a la variación de la carga (*workload*)







El problema del auto-escalado

La carga cambia rápidamente → Automatizar el proceso

Aspectos a resolver:

- Cómo auto-escalar
- Cuándo auto-escalar
- Qué se usa para auto-escalar
- Cuál es el objetivo del auto-escalado

¿Cómo auto-escalar?

Decidir como añadir o quitar recursos, según sea necesario

Se puede realizar a dos niveles:

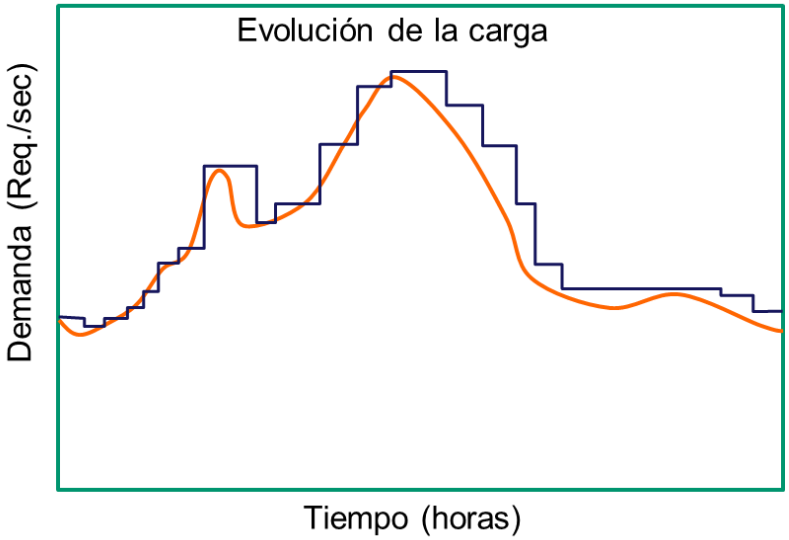
- Nivel de contenedor:
  - Escalado Horizontal (HPA)
  - Escalado Vertical (VPA)
- Nivel de máquina virtual:
  - Escalado Homogéneo (CA)
  - Escalado Heterogéneo (CA-NAP)

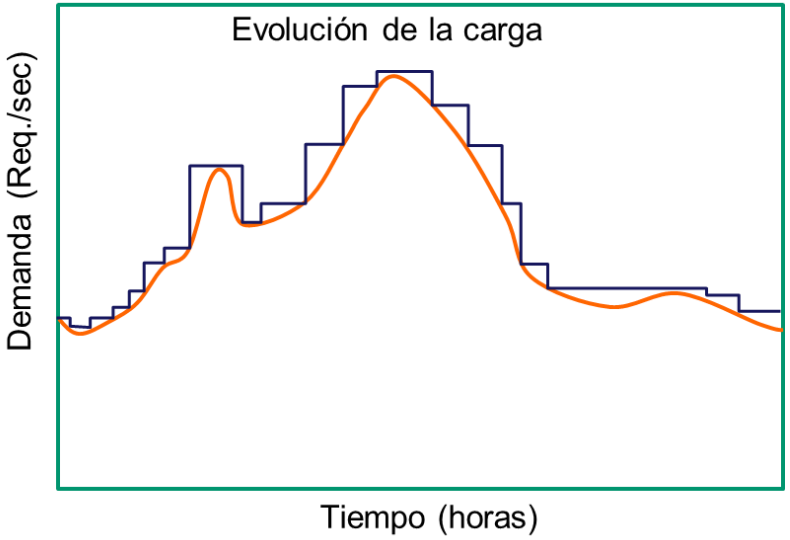
Co-Escalado

¿Cuándo auto-escalar?

Establecer cuándo se toma la decisión de escalar

- Reactivo
- Proactivo o Predictivo





¿Qué se usa para auto-escalar?

Establecer en que se basa la decisión de escalar

Métodos **reactivos** → basados en métricas o reglas

- Utilización de la CPU
- Utilización de la CPU y la memoria
- Utilización de recursos y tiempo de respuesta

Métodos **predictivos** → basados estimaciones o aprendizaje

- Series temporales
- Machine learning

¿Cuál es el objetivo del auto-escalado?

Para qué se lleva a cabo el auto-escalado

Básicamente dos objetivos:

- Minimización del coste
- Cumplir objetivos de QoS (tiempo de respuesta)

Ejemplo de despliegue en el cloud

NETFLIX

Es una empresa cuyo principal servicio es la distribución de contenidos audiovisuales a través de una plataforma en línea o servicio de video bajo demanda por *streaming*

Más de 180 millones de suscriptores, en más de 200 países

Un poco de historia

- Comienza en 1998 con un servicio web de alquiler de DVD
- En 2007 comienza a distribuir vídeo bajo demanda en USA
- En 2008 fallo de su centro de datos → 3 días caído
  - ➢ Migración cloud público (AWS)
    7 años en realizarlo
  - ➢ Reemplazar aplicaciones monolíticas con microservicios

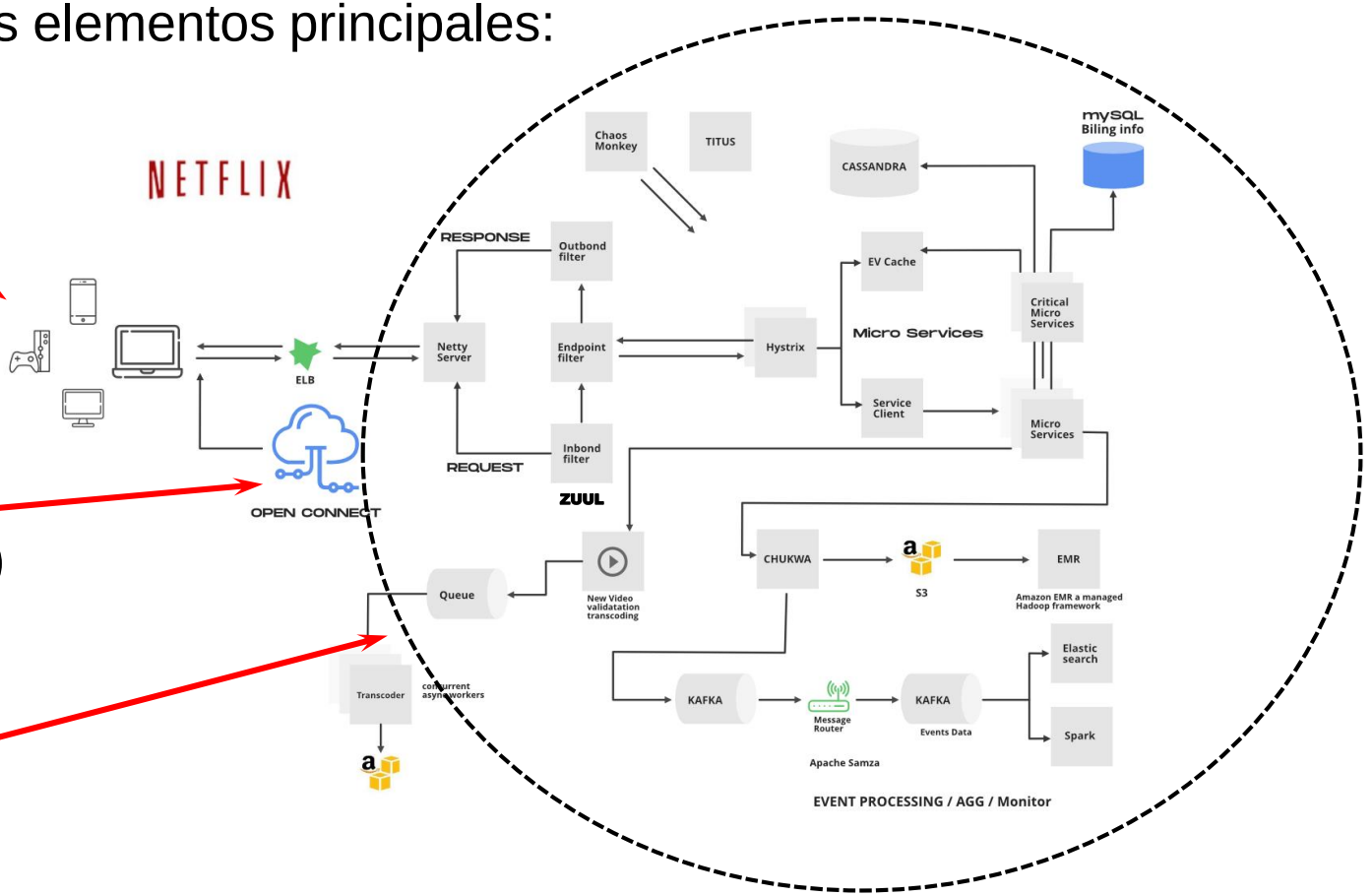
Netflix - arquitectura de alto nivel

Existen tres elementos principales:

Clientes

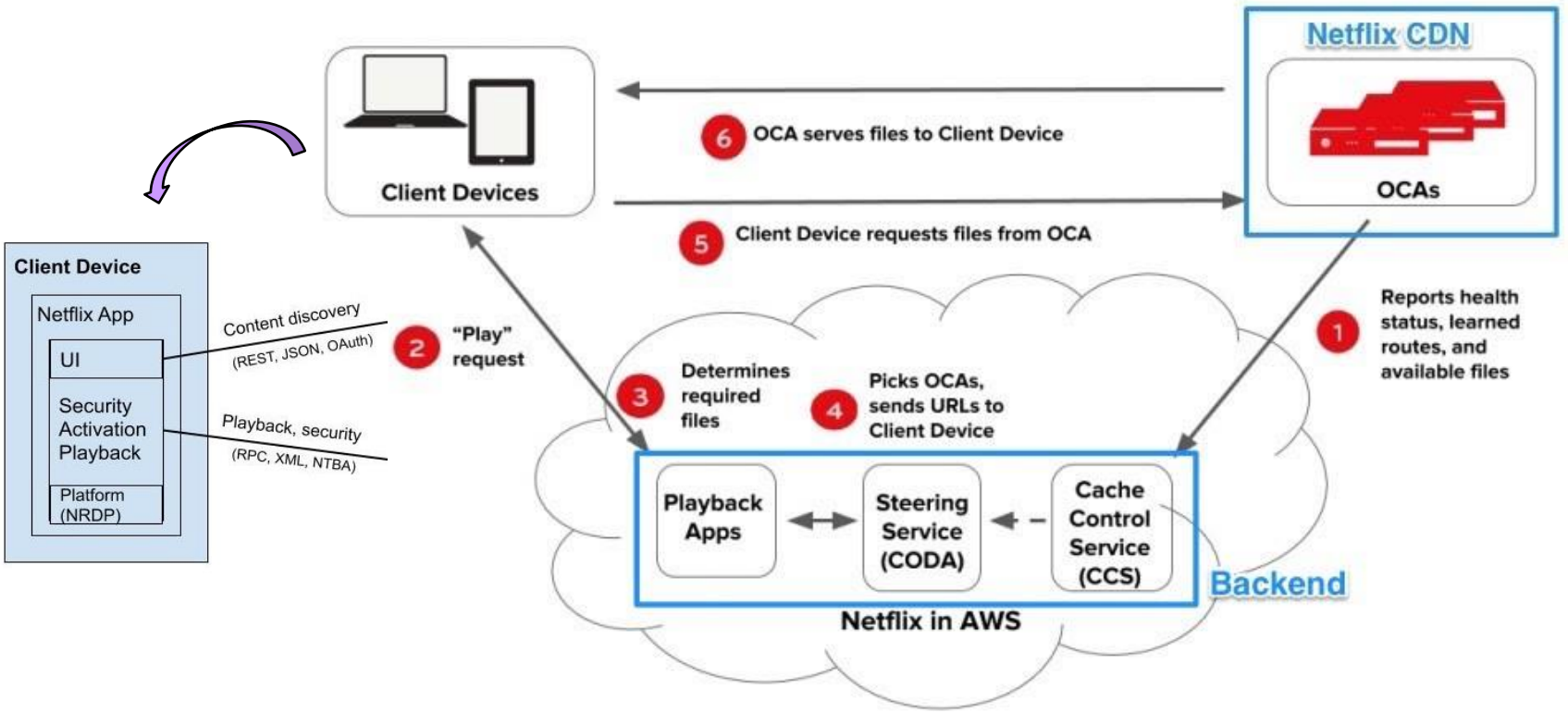
Netflix CDN
(Open Connect)

Backend

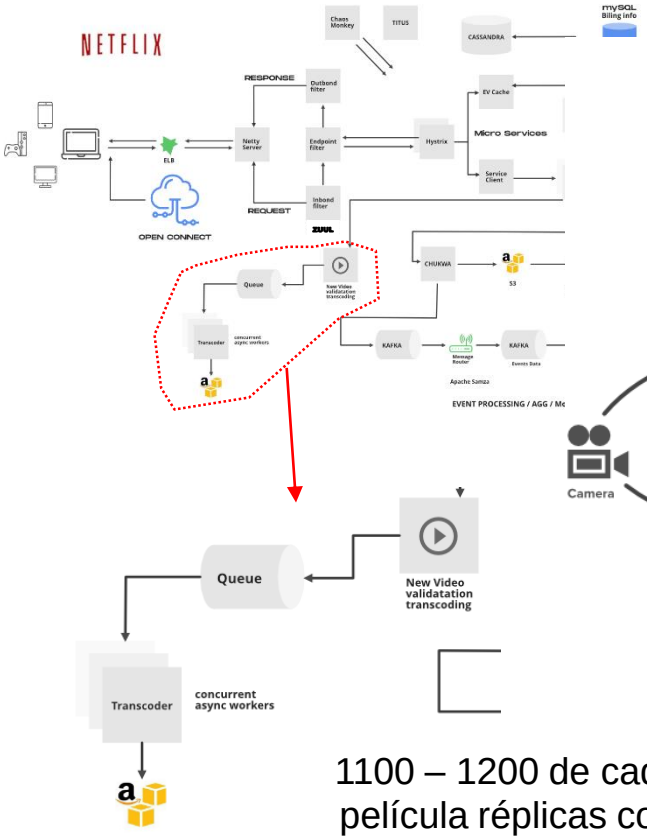


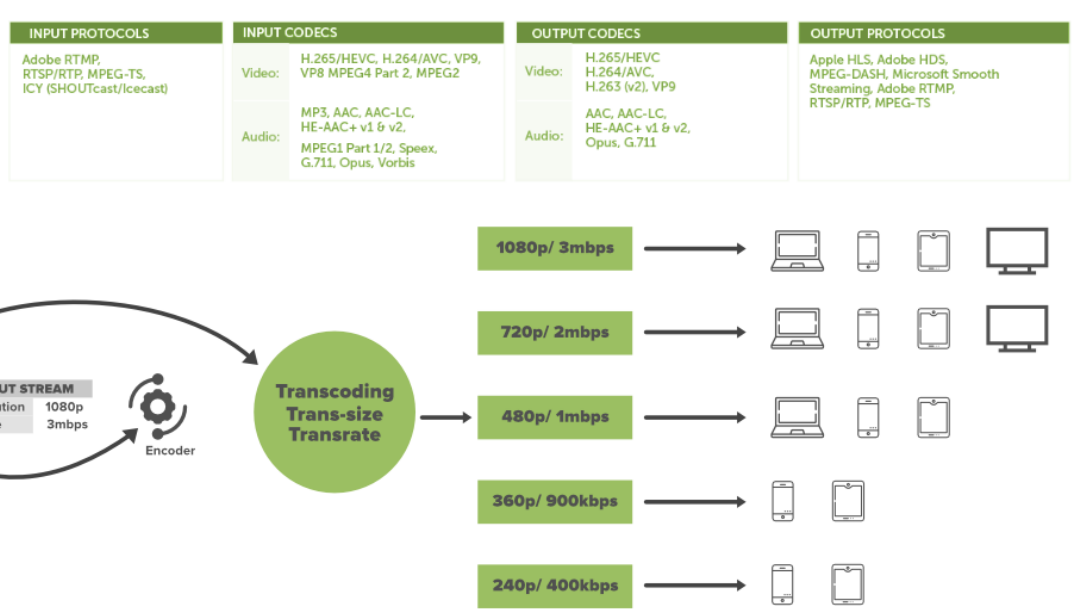
Interacción del cliente

El cliente, a través de una App específica, interacciona con el *backend* para obtener el contenido desde el CDN



Tratamiento de los contenidos



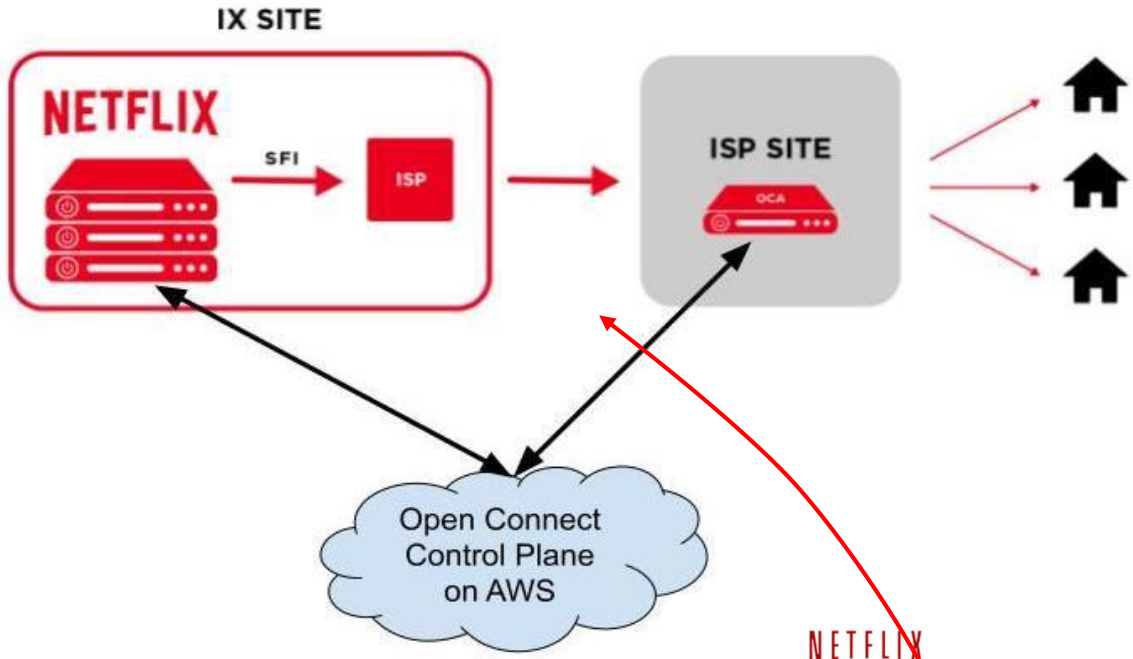


1100 – 1200 de cada película réplicas con diferentes resoluciones

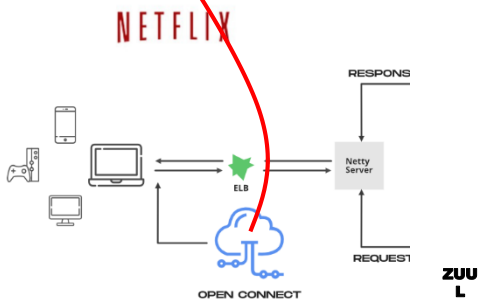
Netflix CDN y Open Connect Apliance (OCA servers)

Proporciona los contenidos de vídeo en la resolución adecuada

Los vídeos se sirven desde servidores especializados llamados Open Connect Appliances (OCAs)



Netlix tiene acuerdos con proveedores de internet (ISPs) y puntos de intecambio de internet (IXPs) distribuidos por el mundo, para situar los servidores OCA desde los que se sirven los vídeos en streaming a los clientes

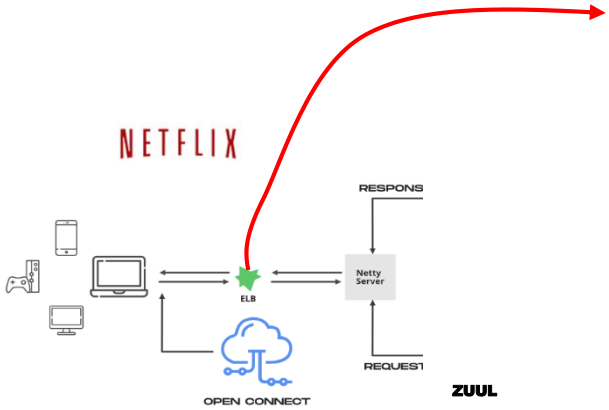


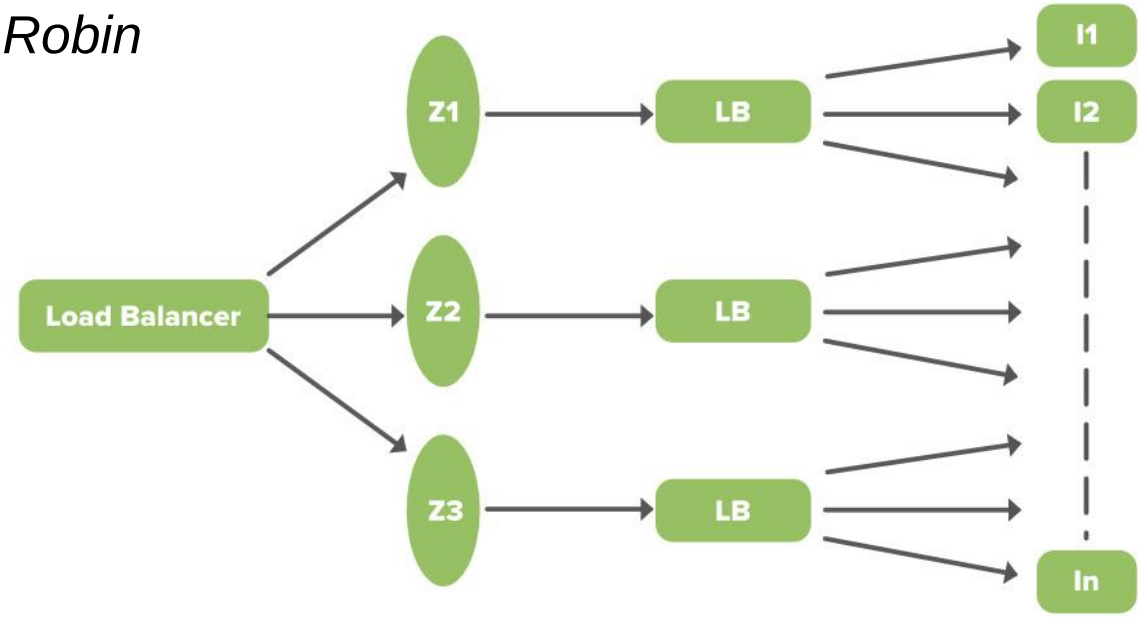
Balanceo de carga

Usa *Elastic Load Balancer* de AWS, trabaja a dos niveles:

1. Entre las tres regiones AWS
2. Entre las máquinas de cada región

Balanceo tipo *Round Robin*

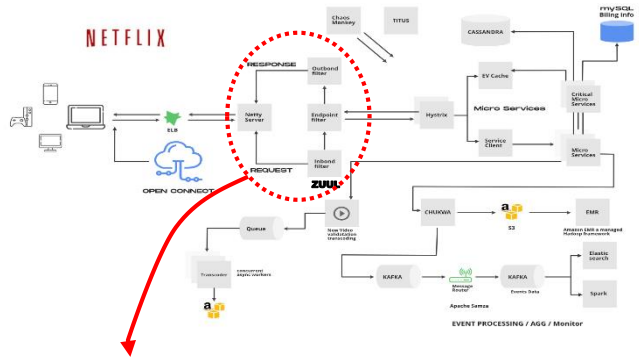


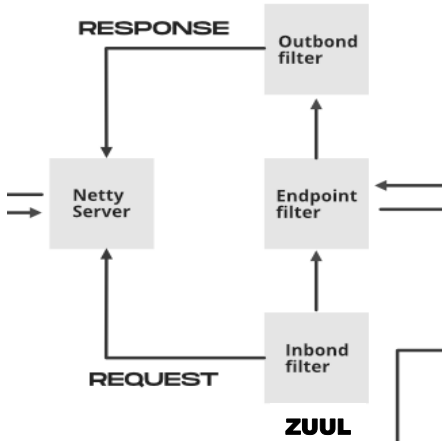


Puerta de entrada: Zuul

Realiza un primer procesamiento de las peticiones de entrada, basada en su url, o path de acceso

- Autenticación
- Enrutamiento a los servicios adecuados
- Filtros del usuario
- Devuelve una lista de OCAs al cliente
- Monitorización y control





Arquitectura de microservcios

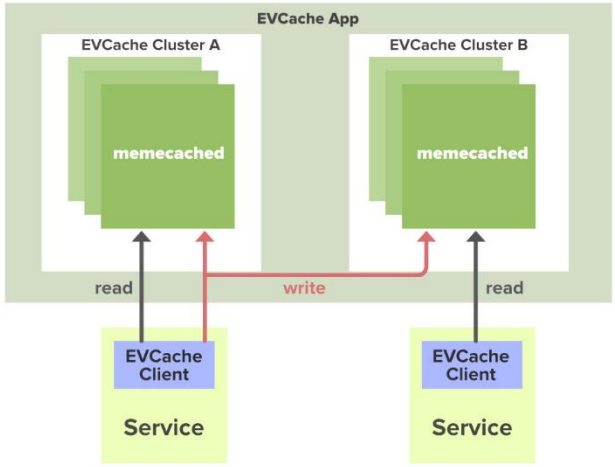
Ejecutan el “core” de la aplicación

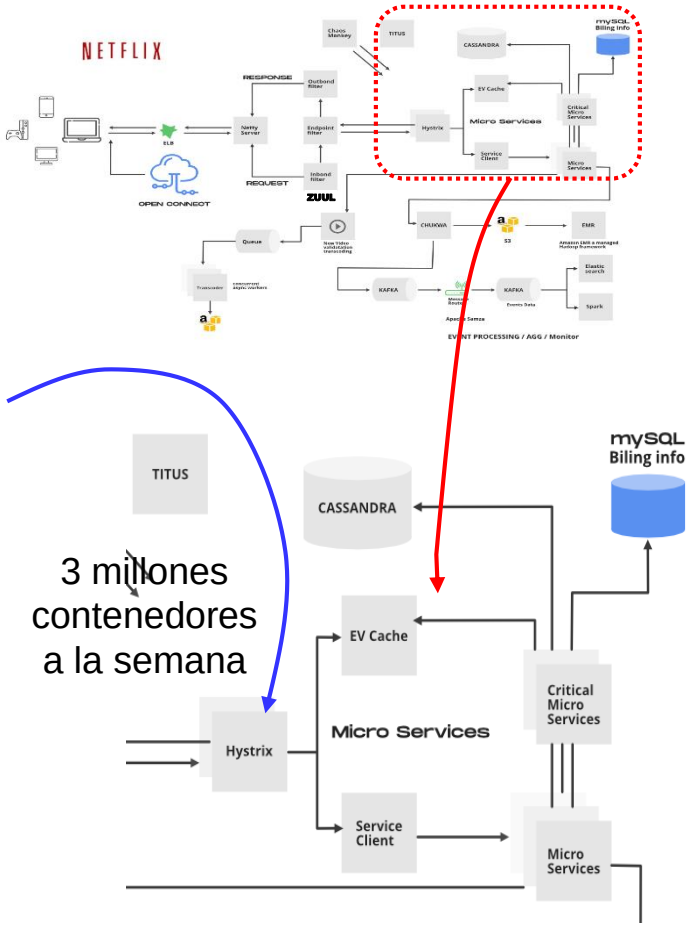
Son elementos sin estado

Separa procesos críticos

Hystrix → controla el tiempo de ejecución

EV cache → rapidez, reduce latencia



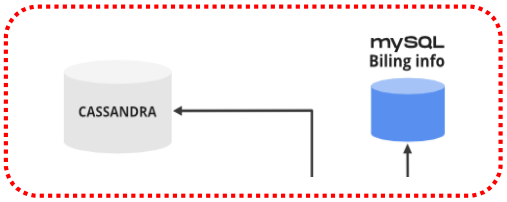


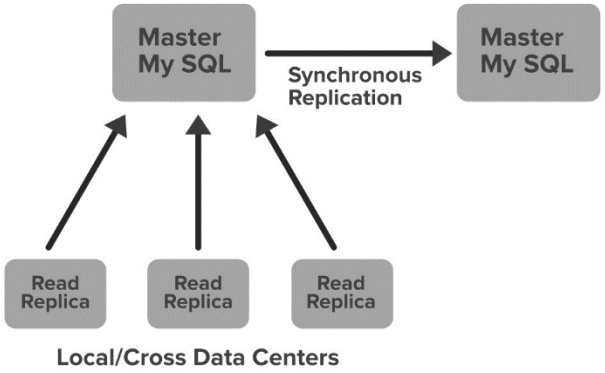
Almacenamiento

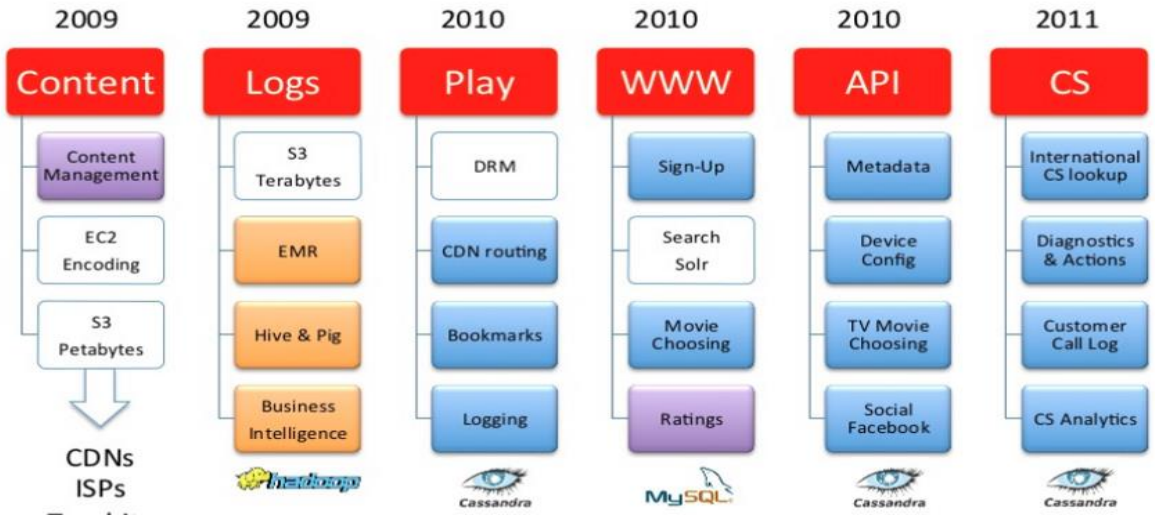
Se utilizan varias soluciones para almacenar información:

Cassandra noSQL

Información visualizaciones del usuario







Proceso de *Billing* en MySQL

Transacciones ACID

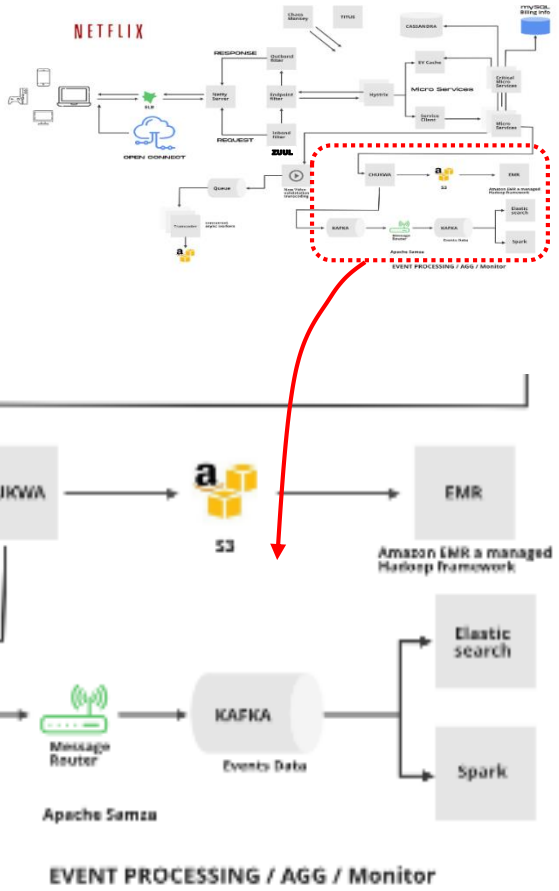
Base de datos maestra replicada

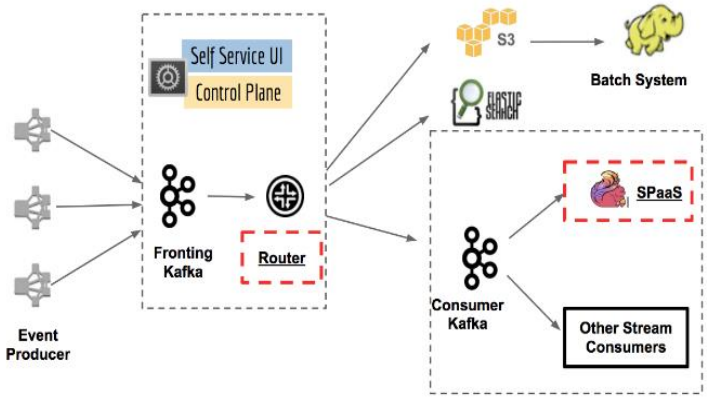
Procesamiento de datos

Recoge y procesa información (data analytics)

- Logs de error
- Actividad de usuario
- Performance
- Actividad de visualización
- Eventos de problemas y diagnósticos

Lleva a cabo recomendaciones al usuario





Conclusión

Netflix ha desarrollado una plataforma que busca:

- Alta disponibilidad
  - Balanceadores de carga
  - Hystrix controla los microservicios, evita fallos en cascada
  - Uso y replicación de cachés
  - Microservicios sin estado
  - Despliegue en tres regiones de Amazon
- Escalabilidad → Usa Auto Scaling de Amazon
- Baja latencia → Conexión al proveedor más rápido y resolución
- Resiliencia
  - Ante fallos de la red y cortes del sistema
  - Chaos engineering → provocan fallos pseudoaleatorios (incluiso caída de una región)

Más información

https://www.geeksforgeeks.org/system-design-netflix-a-complete-architecture/

https://medium.com/swlh/a-design-analysis-of-cloud-based-microservices-architecture-at-netflix-98836b2da45f