upna
Universidad Pública de Navarra
Nafarroako Unibertsitate Publikoa

La nube: AWS

GTIO 2024/2025

Javier Bermejo Razquin

Introducción

1. On Premise “de toda la vida” vs Cloud
2. Modelos de la nube: IaaS, PaaS, SaaS
3. DevOps (nuestro trabajo)

On Premise “de toda la vida”

On Premise “de toda la vida”

- **Dentro del servidor** y la infraestructura de la empresa
- **Modelo tradicional** de aplicaciones empresariales
- Empresa responsable de seguridad, disponibilidad y gestión del software

Conceptos de OnPremise

- **CPD**: Centro de Procesamiento de Datos
- Servidores: máquinas que van a correr SOs, máquinas virtuales, aplicaciones…
- **Capa de networking**: cableado de red, switches, salida a Internet…
- Virtualización: utilización del software para imitar características y comportamiento de un hardware concreto
- **NAS**: Dispositivo de almacenamiento conectado a la red (backup)
- SAI: Sistema de Alimentación Ininterrumpida, sirve para proteger de caídas de corriente

CPD

- **Ubicación geográfica:**
  - Coste económico: coste del terreno, impuestos municipales, seguros, coste eléctrico, coste de datos, etc.
  - Infraestructuras disponibles en las cercanías: energía eléctrica, carreteras, acometidas de electricidad, centralitas de telecomunicaciones, bomberos, etc.
  - Riesgo: posibilidad de inundaciones, incendios, robos, terremotos, etc.

- **Recursos internos:**
  - Falsos suelos y falsos techos
  - Cableado eléctrico y de datos redundante
  - Sistemas de alimentación ininterrumpida
  - Refrigeración
  - Sistema anti-incendios: no agua, protección de equipos

- **Recursos externos:**
  - Múltiple acometida eléctrica y de datos
  - Seguridad en los accesos
  - Medidas de seguridad en caso de incendio o inundación: drenajes, extintores, vías de evacuación, puertas ignífugas, etc.

CPD

- **CPDs duplicados**
  - Un centro de respaldo es un centro de procesamiento de datos (CPD) específicamente diseñado para tomar el control de otro CPD principal en caso de contingencia

- **Racks**
  - Anchura estándar 19” (48,26cm)
  - Profundidad y altura no normalizada
  - Altura medida en U=unidades de rack, 1,75” = 4,44cm
    - Encontramos equipos en formato de rack con altura de 1U, 2U, 3U… según el espacio que requieran
    - La electrónica de red también sigue este estándar

CPD

- **Virtualización**
  - Una máquina física puede correr múltiples máquinas virtuales (VM)
  - Las aplicaciones corren sin modificaciones sobre las VM
  - Las VMs pueden migrar de una máquina física a otra según necesidades
  - Permiten mayor flexibilidad y rapidez para la creación y actualización de servicios
  - Existe hardware específico para poder correr múltiples instancias de VMs de forma más eficiente: Servidores blade
  - Los servicios de computación en la nube suelen basarse en esquemas de virtualización

Problemas de las soluciones On Premise

- **Dependencia** del proveedor
- Coste oculto: coste de oportunidad
- **Riesgo** de la conexión a Internet
  - Abrir los servidores On Premise
  - Exigencia de movilidad por parte de clientes
- Flexibilidad y escalabilidad muy limitada
- Alto coste de instalación y mantenimiento
- **Vulnerabilidad** de las instalaciones
- Cuello de botella en la conexión

Llega la nube

- En pocas palabras: **cómputo en Internet**

- Frontend / Backend
  - Tu frontend
  - Su backend (alquilado)

- Un concepto no tan reciente
  - Desde que existe internet
  - 1997 → entorno academico

Características de la nube

- No eres propietario del hardware
- **Autoservicio** bajo demanda
- Acceso amplio a servicios
- Pool de recursos dinámicos
- Medición de uso dinámica

Algunos conceptos de la nube

- **Fault tolerance**
  - Habilidad de un sistema de continuar funcionando cuando algo falla
  - Zero Down-Time

- **High Availability**
  - Habilidad de minimizar la pérdida de servicio cuando algo falla
  - N ‘nines’ → %

- **Disaster Recovery**
  - Contra catástrofes
  - Plan de recuperación
  - Puntos y tiempos específicos

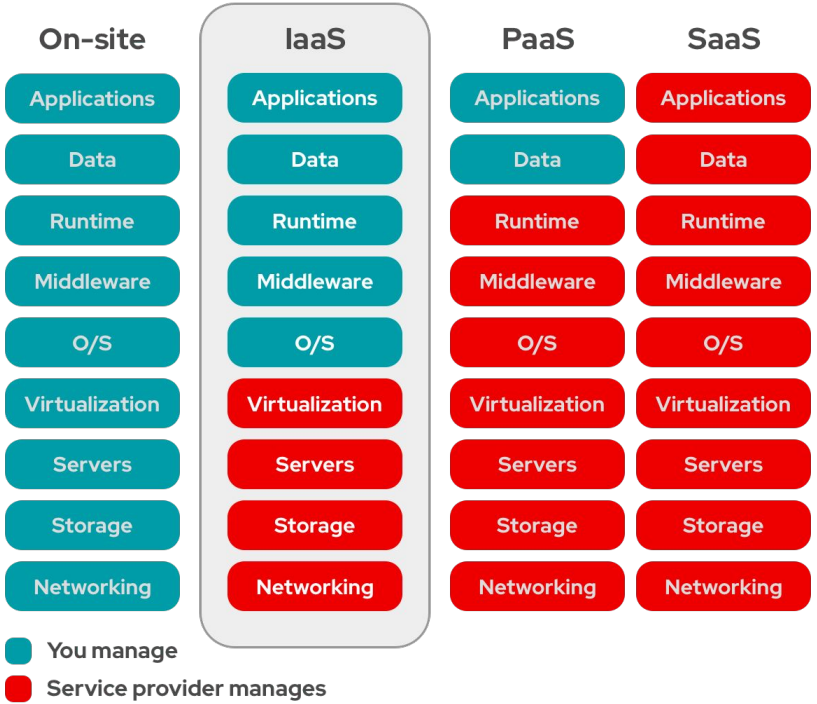
Algunos conceptos de la nube

- **Escalabilidad**
  - Habilidad de escalar la capacidad de cómputo, tanto de forma **horizontal** (número de máquinas) como **vertical** (tamaño de las máquinas)

- **Elasticidad**
  - Capacidad de aumentar o disminuir cualquier recurso con rapidez para responder a la demanda

- **Agilidad**
  - Capacidad de desarrollar, testear e implementar rápidamente software y aplicaciones que aumentan el valor del negocio

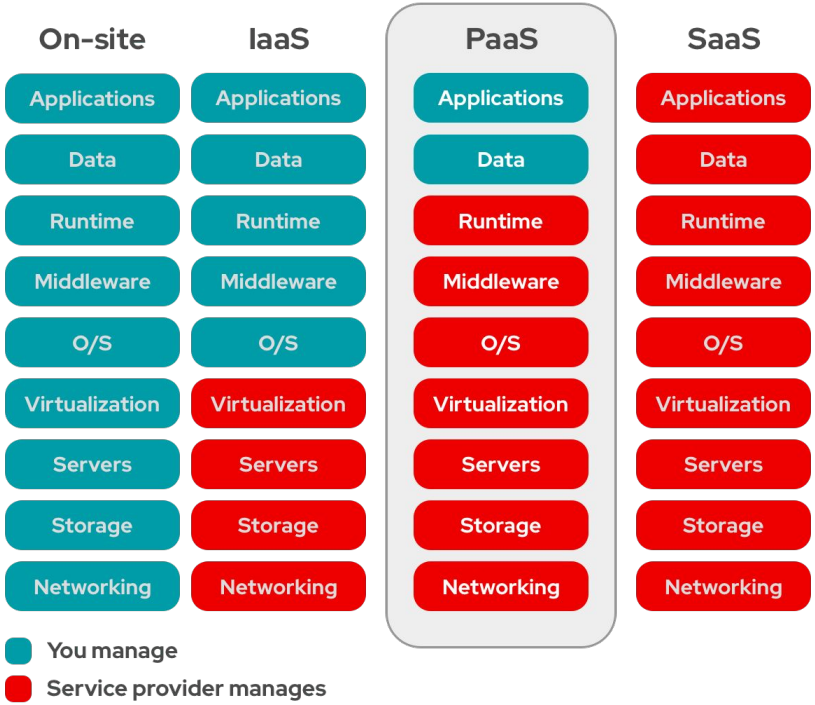
IaaS: Infraestructura como servicio



IaaS: Infraestructura como servicio

- **“Alquilar el Hardware”**
- Máquinas virtuales, firewalls, sistemas de backup, balanceadores de carga…
- Es la base: contiene los bloques fundamentales para la TI en la nube
- Ejemplos:
    - Amazon Web Services (EC2)
    - Microsoft Azure
    - Google Cloud

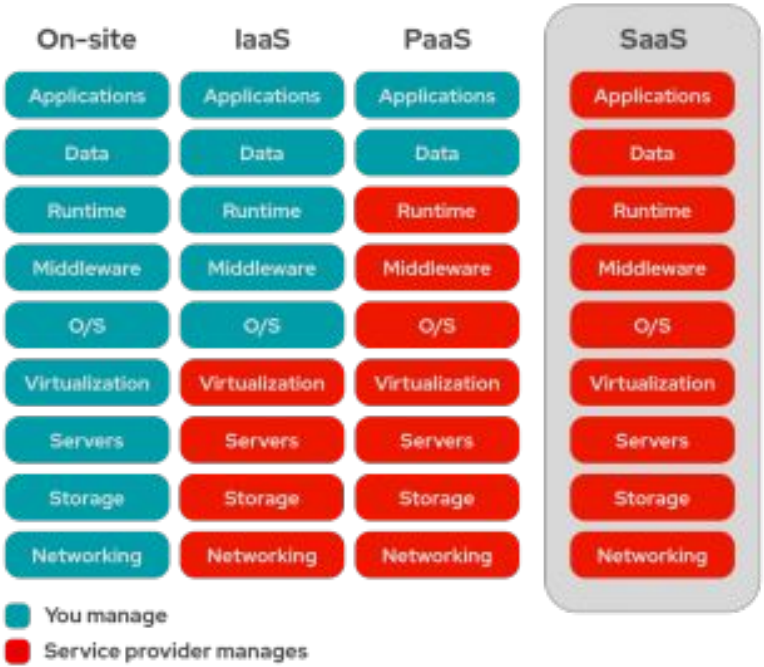
PaaS: Plataforma como servicio



PaaS: Plataforma como servicio

- **Plataformas sobre las que lanzar aplicaciones**, como bases de datos, middleware, herramientas de desarrollo…
- Ideal para desarrolladores que quieren centrarse en sus aplicaciones
    - Sin preocupaciones de los recursos
- Es más complicada de entender (es software, pero no es SaaS)
- Más conocida: Google App Engine
- Otros ejemplos:
    - SAP Cloud
    - Heroku

SaaS: Software como servicio



SaaS: Software como servicio

- El **software** de una empresa está alojado en la nube y **se contrata su uso**.
- Requiere el **mínimo nivel de gestión**
  - Datos
  - Usuarios
- Ejemplos de SaaS:
  - Microsoft Office 365
  - Gmail, Google Drive…

Entonces cuáles son las diferencias entre ellas?

- **IaaS**: Servidores y redes sin configurar. Algo así como un alquiler del material necesario para desarrollar tus programas. Es como si fueses a la cocina de un amigo a hacerte tu la cena.
- **PaaS**: Servidores y redes configuradas. Un software instalado que te permite desarrollar tu proyecto con más facilidad. Es como si usases la cocina de tu amigo y este también te ayudase a hacer la cena.
- **SaaS**: Servidores y redes configuradas. Un software instalado que te soluciona parte o todo tu proyecto. Como si tu amigo te invitase a su casa a cenar.

IaaS: Infraestructura como servicio

Aspectos que deben considerarse a la hora de elegir un proveedor de IaaS

- **Flexibilidad:** adquiera únicamente los elementos que necesita para su caso práctico y amplíelos o redúzcalos en función de las necesidades empresariales.
- **Asequibilidad:** la IaaS es una opción asequible, ya que los gastos generales son bajos y no genera costos de mantenimiento.
- **Control:** el usuario controla la infraestructura.
- **Seguridad:** ¿tiene el proveedor una reputación confiable y posee los recursos necesarios para evitar y gestionar las amenazas de seguridad? ¿Existen protocolos documentados de recuperación ante desastres que garanticen la continuidad empresarial?
- **Servicio**: ¿cuáles son los acuerdos de nivel de servicio (SLA) del proveedor?
- **Confiabilidad:** el rendimiento y la velocidad dependen en gran medida del proveedor.

SaaS: Software como servicio

Aspectos que deben considerarse a la hora de elegir un proveedor de SaaS

- **Interoperabilidad:** La integración con tus otros programas tiene que ser posible. Si el SaaS no tiene una solución pensada, la integración puede ser más compleja.
- **Vendor lock-in:** Bloqueo de proveedor. Es fácil unirse a un servicio y difícil salir de él. Empezar a usar un SaaS puede crear una dependencia con este.
- **Seguridad:** Al utilizar un servicio externo cedes tu información a otra empresa. Eso puede ser un problema por temas de seguridad y/o confidencialidad
- **Falta de personalización:** En un SaaS no puedes decidir futuras funcionalidades. Si necesitas algo y tu proveedor no lo ofrece, solo te queda esperar (o pedirlo)
- **Rendimiento:** Obtienes el rendimiento que te ofrezcan. No puedes mejorarlo por tus propios medios, así que debes adaptarte a él

PaaS: Plataforma como servicio

Aspectos que deben considerarse a la hora de elegir un proveedor de PaaS

- **Básicamente una mezcla de los dos anteriores**
- Como hemos dicho el PaaS es un punto medio entre los otros dos servicios, y sus ventajas/inconvenientes son un punto medio entre los dos
- Los PaaS no tienen una características tan claras como los IaaS o los SaaS, por lo tanto sus ventajas e inconvenientes tampoco lo serán
- Deberíamos estudiarlo caso por caso

Tener software en la nube no es una tarea fácil

- Hace falta tener conocimientos en redes, servidores y bases de datos para poder crear la base donde desplegar el software
- Necesitas un equipo que se encargue de desplegar estas aplicaciones con éxito y sin disrupción.
    - Pueden ser equipos de solo nube
    - O enseñar a tus empleados a usar la nube e ir aprendiendo poco a poco
- Hay que **mantener la nube**. Al fin y al cabo los servidores son máquinas en producción y estas pueden fallar. La monitorización es muy importante
- La nube puede llegar a costar mucho dinero si no se optimiza. Es necesario saber escalar y optimizar los despliegues
- Y mientras tanto otros equipos necesitarán ayuda para adaptar el software a la nube

Y.. quien hace todo esto?

- Los DevOps/SRE/Platform Engineers
- Somos una especie de mezcla de IT, desarrolladores, arquitectos y operaciones
- Es un puesto muy creativo y con muchas funciones y responsabilidades.
- Tenemos muchísimas funciones y responsabilidades, pero a un nivel más de “barro”
- **Interactuamos con muchos equipos y aprendemos muchísimo sobre tecnologías**

DevOps?

- Realmente los DevOps no existen, pero es una manera fácil de entender la manera de trabajar de la gente que se dedica a la nube.
    - Platform Engineer, Build Engineer, Release Manager, Site Reliability Engineer (SRE), Data Analyst, etc.
- Nuestras funciones son las de **crear**, **mantener** y **optimizar** todo lo que tenga que ver con infraestructura en la nube. Y a veces incluso explicar a la gente como hacerlo!
- Nuestras tareas más comunes: desplegar nuevos microservicios, actualizar estos, mantener la nube con pequeñas tareas, optimizar la nube para reducir gastos y monitorear que todo funcione bien

Explicar a la gente como hacerlo?

- “Y a veces incluso explicar a la gente como hacerlo!”
- La frase suena muy pretenciosa, y suena un poco a actitud de “arquitecto de software”
- Soy fiel defensor de que cada desarrollador tiene que ser libre de adaptar su código a su gusto y crear su propia infraestructura
- Pero eso no quita que tenga que ser consciente de las limitaciones en producción, y puede ser muy dificil transmitir ese conocimiento cuando los equipos están separados y no se encargan de operar ese producto
- Cada caso es único y lo mejor que se puede hacer en estos casos es hablar entre los equipos, aunque la experiencia es un grado y hay que escuchar muy bien a un equipo operativo (o te acabarán llamando cuando pasen las cosas)

Y como lo hacemos?

- **Automatizando**.
- **Todo**.
- Absolutamente todo.
- Algo no automatizado en producción va a ser nuestra mayor pesadilla.
- Y no lo hacemos por hacer. Si algo va a producción sin estar automatizado y falla podemos dejar a nuestros clientes sin servicio por mucho rato.
- Además sin automatizar somos más propensos a tener **errores**.
- Son sistemas muy grandes y complejos y si algo falla encontrar el error puede costar mucho tiempo
- En general la automatización en infraestructura viene a ser como los tests unitarios en código.
    - Necesarios, aunque el coverage no sea siempre el 100%
    - Y si no están… huele mal

Razones para automatizar

- Si algo falla, lo **arreglamos** mucho más **rápido**
- Cometeremos **menos errores**
- Automatizar es más complicado que hacerlo a mano, lo que nos ayuda a comprender mejor todo y estar preparados para futuras incidencias
- **No somos dependientes** de quien ha realizado el software.

Razones para no automatizar

- **Se tarda muchísimo más** en automatizar que en hacerlo manualmente.
- Un nuevo desarrollo puede que no tenga éxito y acabe abandonando.
- **Agilidad**

Herramientas básicas para automatizar

- **CI/CD**: Continuous Integration and Continuous Deployment. Integración y despliegue continuo
- **Tests**: Diferentes suites de tests para confirmar que todo funciona como debe
- **Métricas**: Nos indican cómo está respondiendo el sistema
- **Calidad**: Tenemos que construir una infraestructura de calidad para poder automatizar correctamente

Las incidencias

- Va a haber **incidencias**. Acéptalo.
- Y no hay que sentirse mal por ello, si no **aprender** de los errores.
- Es muy importante que después de una incidencia el equipo se reúna y estudie que ha pasado y cómo puede evitar el error.
- Nos podemos preparar para estas incidencias:
    - **Chaos Monkey**: Netflix saboteándose a si mismo.
    - **Disaster Recovery**: Como hacemos si todo sale mal.