



Universidad de Oviedo



Disposición (Allocation)

Curso 2018/2019



Jose Emilio Labra Gayo

Disposición

Relación del software con el entorno

¿Dónde se ejecuta cada componente?

¿Cómo se envía el software?



Disposición

Vista de despliegue

Empaquetamiento, distribución, despliegue

Canales de distribución

Opciones de entrega

Entorno de ejecución

Deployment pipeline

Software en producción

Configuración

Planificación de capacidades

Logging & Monitorización

Incidentes y post-mortem

Ingeniería del caos

Punto de vista de despliegue

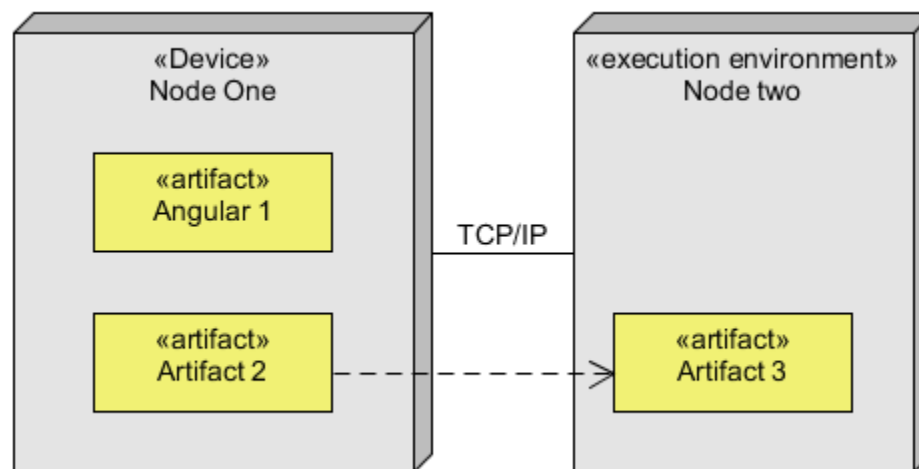
UML proporciona diagramas de despliegue

Artefactos asociados con nodos computacionales

2 tipos de nodos:

Nodo dispositivo (Device)

Nodo de entorno de ejecución



Empaquetamiento, distribución y despliegue

Empaquetamiento

Crear ejecutable a partir del código fuente

Consiste en:

Código compilado

Incluso para lenguajes interpretados (Javascript):

Transpiled, ofuscado & minimizado

Ficheros de configuración

Variables de entorno

Credenciales, etc.

Librerías & dependencias

Manuales de usuario y documentación

Scripts de instalación



Publicación de releases

Una *release* supone cambios de funcionalidad

Planificación

Publicar una release implica costes

Normalmente, los usuarios no quieren nuevas releases

Factores externos:

Marketing, clientes, hardware, ...

Modelo ágil: *releases* frecuentes

Entrega continua minimiza el riesgo

Entrega continua

Continuous delivery

Publicar releases frecuentes para obtener feedback tan pronto como sea posible

Canal de despliegue (*Deployment pipeline*)

Ventajas:

Abrazar el cambio

Minimizar riesgos de integración



Filosofía Wabi-sabi

Aceptar la imperfección

Software no terminado: Suficientemente bueno

Feature toggles y canary releases

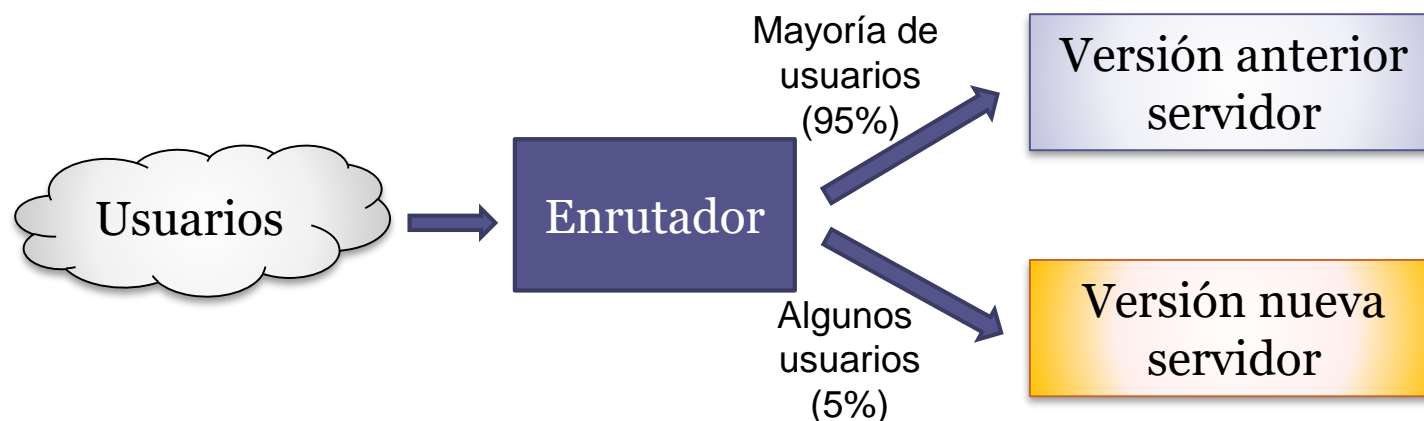
Feature toggles

También conocidos como *feature flags*, *feature bits*,...

Modificar comportamiento del Sistema sin modificar el código

Canary releases

Introducir nuevas versiones mostrando cambios lentamente a un subconjunto de los usuarios



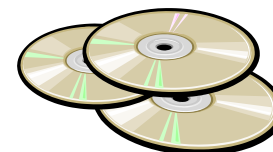
<https://martinfowler.com/articles/feature-toggles.html>

<https://martinfowler.com/bliki/CanaryRelease.html>

Canales de distribución

Distribución tradicional

CDs, DVDs, ...



Basada en Web

Descargas, FTP, ...



Mercados de aplicación

Paquetes Linux

Almacenes de aplicaciones

AppStore,

Google Play,

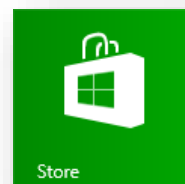
Windows Store



App Store



Google play



Store

Opciones de ejecución de software

Centro de datos (*On-premises*):

Se instala y ejecuta en computadores del cliente

Cloud computing: SaaS (Software as a Service)

Se alquilan capacidades computacionales

Edge computing

Computación cerca de dispositivos finales

Dispositivos conectados procesan datos cerca de donde los datos se crean

Example: IOTs, coches conectados, ...

Fog computing

Computación en nodos intermedios de la red local

Opciones de ejecución de software

Capa Cloud

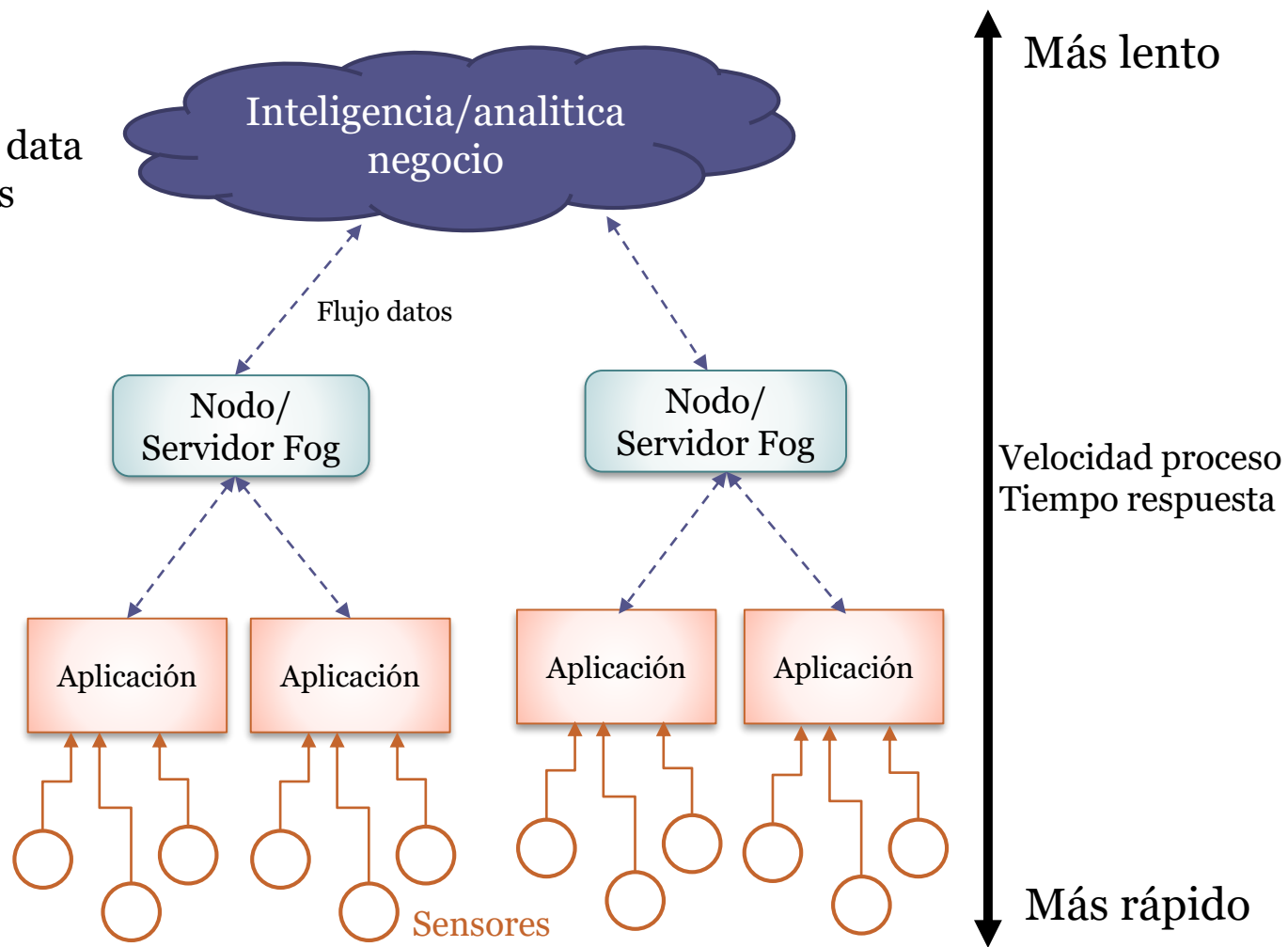
Procesamiento Big data
Almacenes de datos

Capa Fog

Red local
Respuesta control

Capa Edge

Tiempo real
Micro almacén datos
Visualización *On-premises*
Sistemas empotrados



Entornos de ejecución

Máquinas físicas

Computador grande vs granjas de servidores

Máquinas virtuales

Múltiples sist. operativos conviven en misma máquina

Proporcionan portabilidad y aislamiento

Solución muy popular

Mayoría de aplicaciones Web = sobre máquinas virtuales

Rendimiento menos predecible

Contenedores

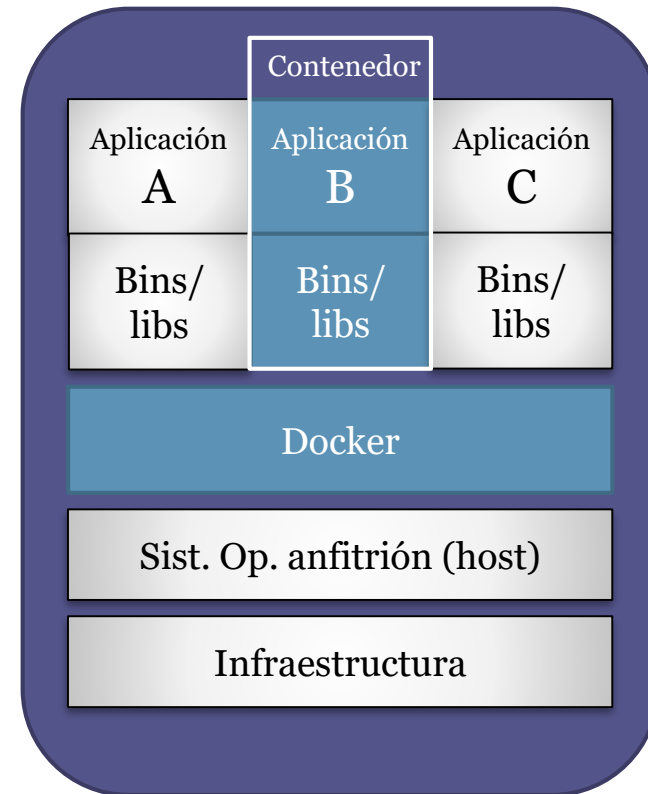
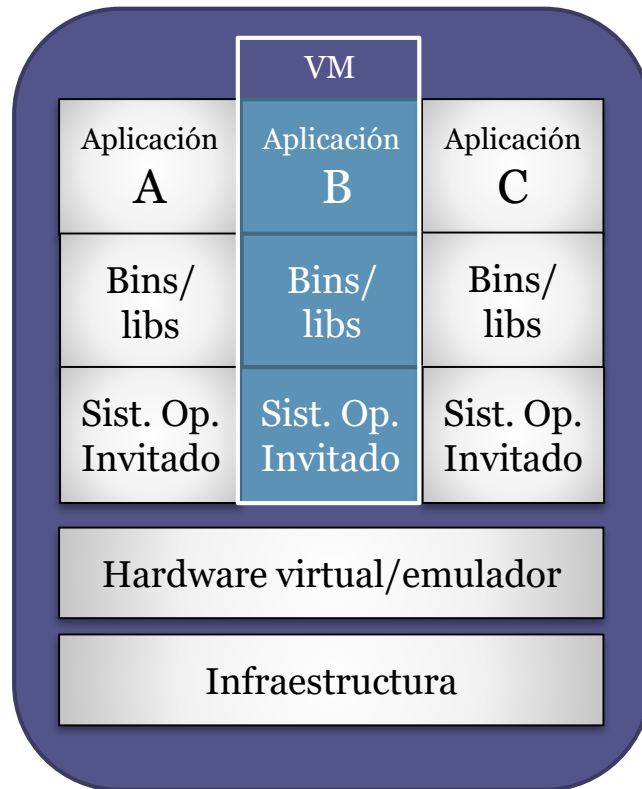
Docker

Ejecución en procesos locales

Distribución de imágenes en contenedores



Máquinas virtuales vs Contenedores



Contenedores

Ventajas

Rendimiento: menor sobrecarga y recursos sistema

Mayor portabilidad

Fácil instalación: Despliegue como código

Se adapta muy bien a microservicios

Retos:

Gestión:

Aplicaciones basadas en contenedores pueden tener muchas instancias

Coordinación entre contenedores

Técnicas: Kubernetes, Docker swarm

Despliegue continuo

Canal de despliegue (Deployment pipeline):
Automatizar el proceso de construcción, pruebas, empaquetamiento y despliegue

Objetivos

- Crear entornos de ejecución bajo demanda
- Resultados rápidos, fiables, repetibles y predecibles
- Entornos de ensayo y producción consistentes
- Bucles de realimentación rápidos
- Alcanzar días de *release* sin riesgos (incluso aburridos)

Deployment pipeline

Patrones

Infrastructure as code

Mantener todo en control de versiones

Código

Configuración y scripts

Datos

Documentación

Alinear desarrollo y operaciones (DevOps)

Herramientas:

Ansible, Chef, Puppet,...

Mejores prácticas: <https://12factor.net/>

Software en producción

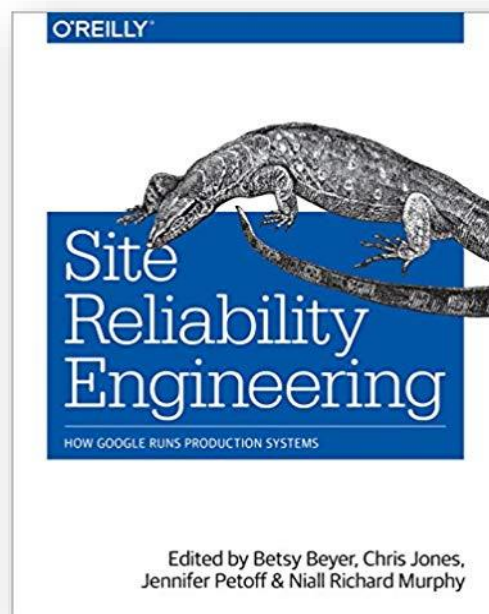
Algunos atributos de calidad:

Disponibilidad

Fiabilidad

Observabilidad

Libros recomendados



Gratis *online*

Fiabilidad

Planificación de capacidad

Pruebas de carga

Ejemplo: JMeter, Gatling

Balanceo de carga

Aumentar fiabilidad mediante redundancia

Failover (commutación por error)



Logging y monitorización

Atributo de calidad: Observabilidad

Normalmente no lo piden los usuarios

Logging

Suele ser fácil de generar

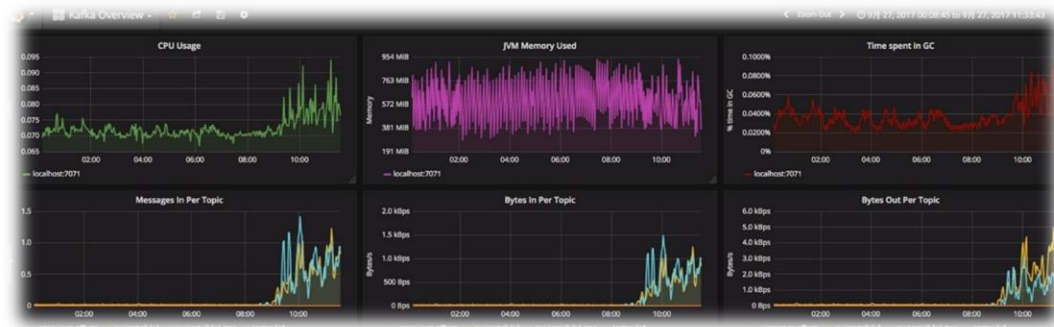
Logging como *stream processing*: Apache Kafka

Métricas & Monitorización

Bases de datos de series temporales y visualizaciones

Prometheus, Graphite, Grafana, Datadog, Nagios, ...

Health checks



Incidentes y post-mortem

Resolver y revisar un incidente

Asegurar que el equipo lo ve sin buscar culpables

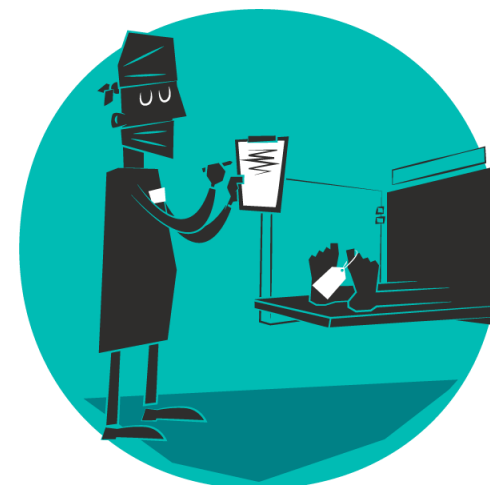
Crear informe *post-mortem*

Detalles del incidente

Línea temporal y acciones tomadas para resolverlo

Análisis de causa raíz (*root cause*)

Identificar medidas preventivas



Ingeniería del caos

Propuesto por Netflix en 2010 (*Chaos Monkey*)

Introducir fallos adrede en los sistemas

Probar sistemas distribuidos

Romper cosas adrede

Pruebas mediante inyección de fallos

Asegurar que el fallo de una instancia no afecta al sistema global

Anti-fragilidad y resiliencia

Capacidad para absorber perturbaciones

<https://github.com/Netflix/chaosmonkey>

Fin