Universidad de Oviedo

ES
Español

Escuela de Ingeniería Informática

Disposición (Allocation)

Curso 2019/2020

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

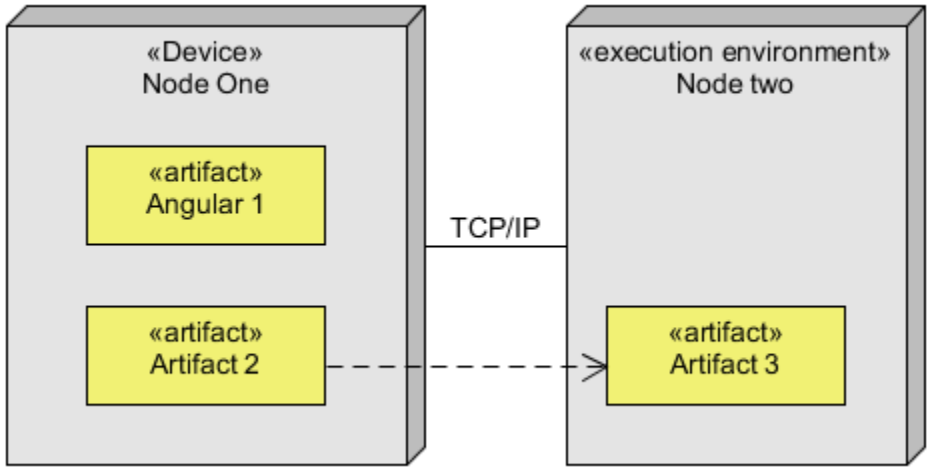
Punto de vista de despliegue

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.
- Librarías & dependencias
- Manuales de usuario y documentación
- Scripts de instalación

Publicación de releases

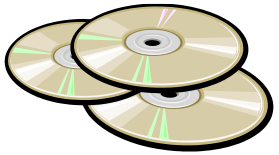
Una *release* supone cambios de funcionalidad

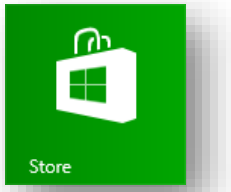
Planificación

- 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

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





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

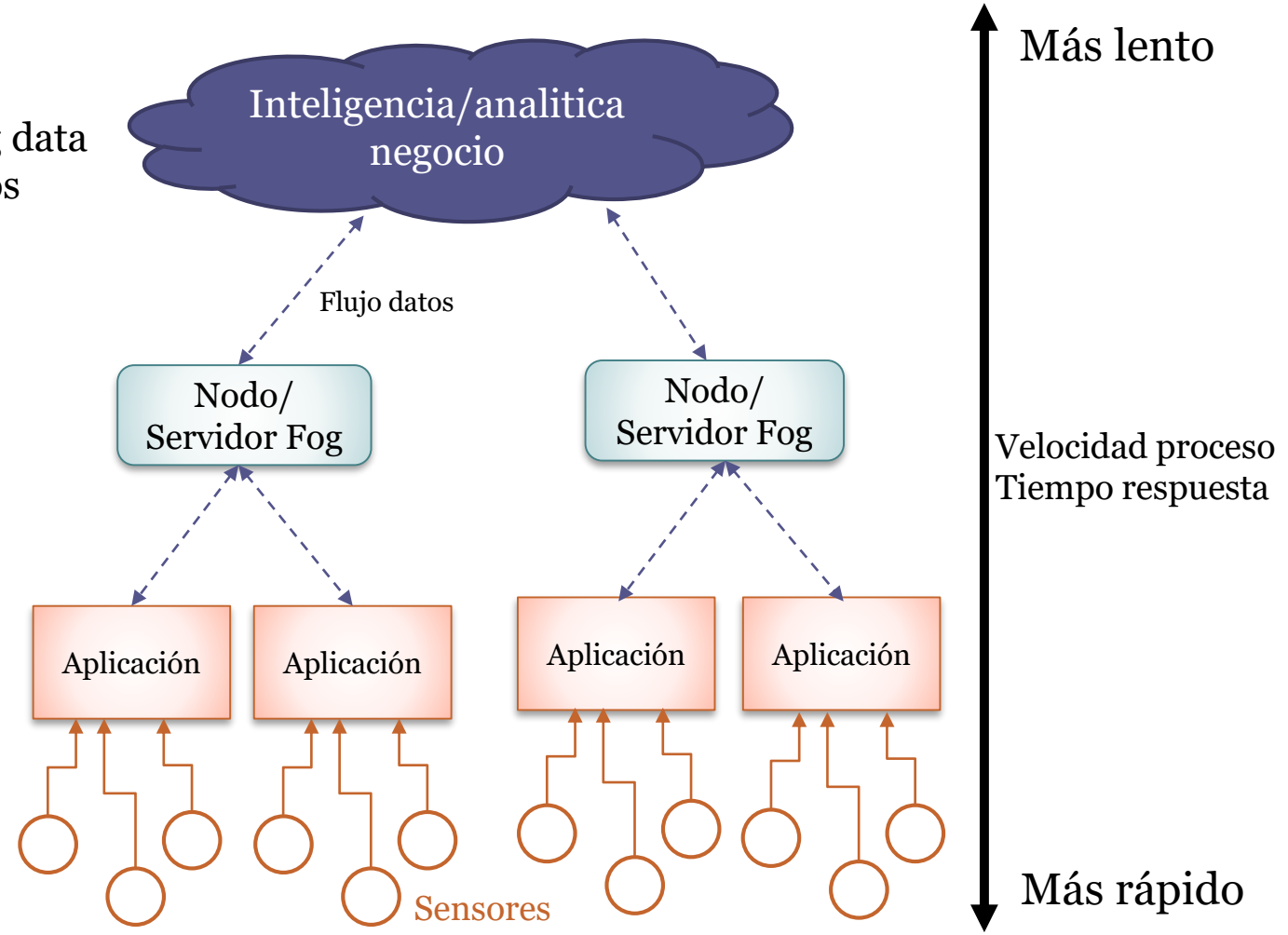
Procesamiento Big data
Almacenes de datos

Capa Fog

Red local
Respuesta control

Capa Edge

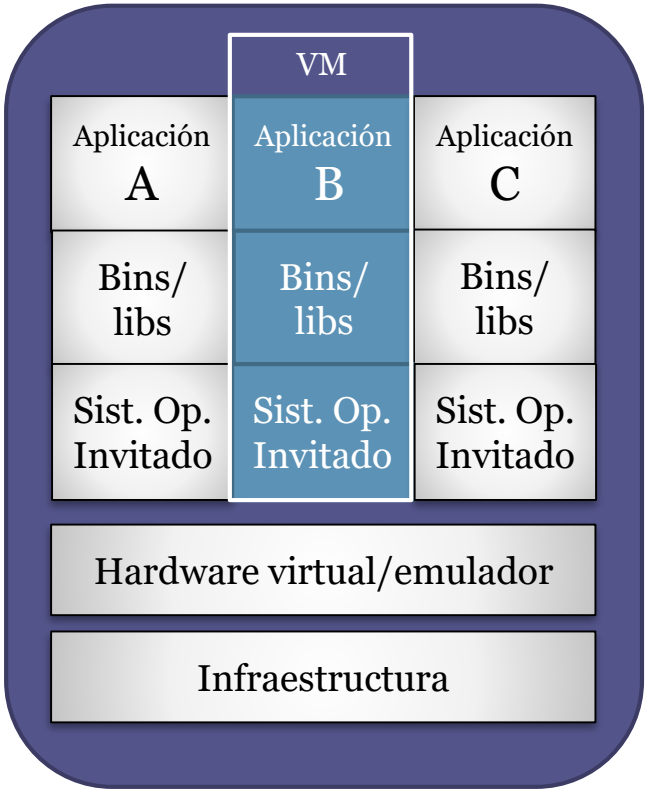
Tiempo real
Micro almacén datos
Visualización *On-premises*
Sistemas empotrados

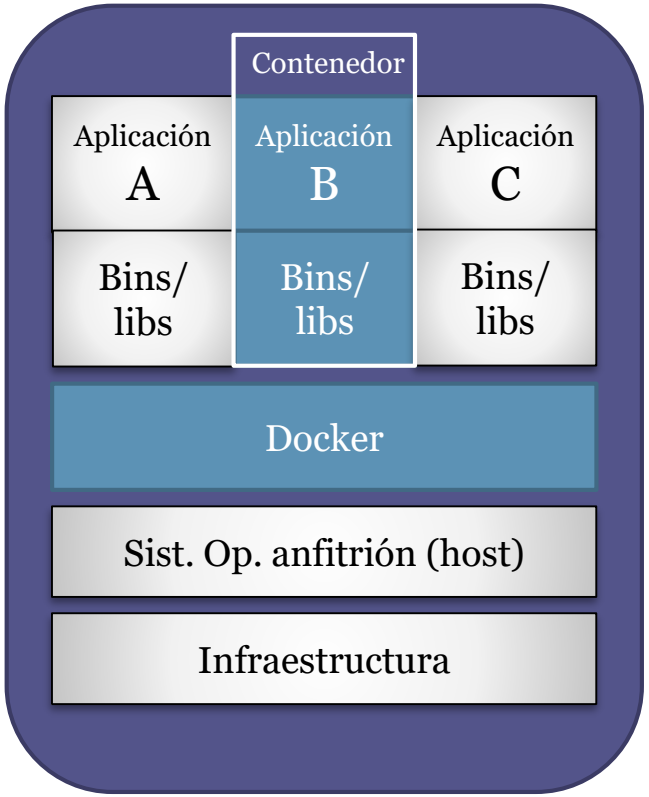


Entornos de ejecución

- 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

Entrega continua

- Continuous delivery
  - Publicar releases frequentes 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

Canal de despliegue

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 (12 factores), siguiente slide

12 factores https://12factor.net/

I. Código base (Codebase): Un código base sobre el que hacer el control de versiones y multiples despliegues

II. Dependencias: Declarar y aislar explícitamente las dependencias

III. Configuraciones: Guardar la configuración en el entorno

IV. Backing services: Tratar a los “backing services” como recursos conectables

V. Construir, desplegar, ejecutar: Separar completamente la etapa de construcción de la etapa de ejecución

VI. Procesos: Ejecutar la aplicación como uno o más procesos sin estado

VII. Asignación de puertos: Publicar servicios mediante asignación de puertos

VIII. Concurrencia: Escalar mediante modelo de procesos

IX. Desechabilidad: Hacer sistema más robusto intentando conseguir inicios rápidos y finalizaciones seguras

X. Paridad en desarrollo y producción: Mantener desarrollo, preproducción y producción tan parecidos como sea posible

XI. Historiales: Tratar historiales como una transmisión de eventos

XII. Administración de procesos: Ejecutar tareas gestión/administración como procesos que solo se ejecutan una vez

Pruebas y entrega continua

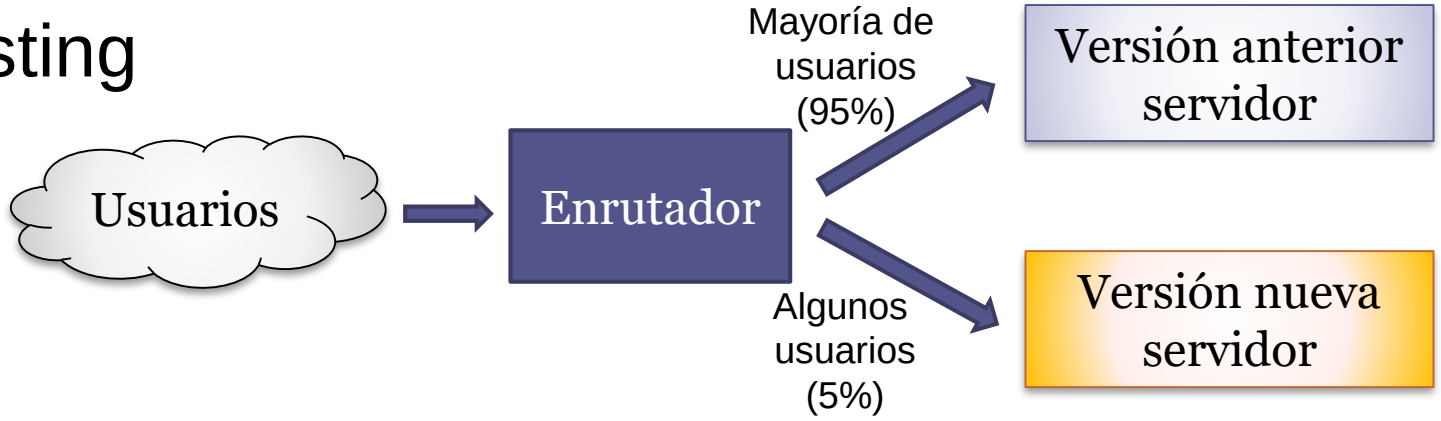
Feature toggles

- 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

A/B testing



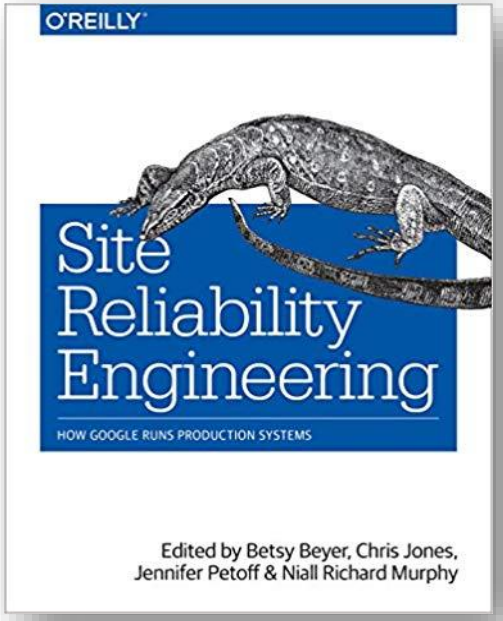
https://martinfowler.com/articles/feature-toggles.html
https://martinfowler.com/bliki/CanaryRelease.html

Software en producción

Algunos atributos de calidad:

- Disponinbilidad
- 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

https://landing.google.com/sre/sre-book/chapters/postmortem-culture/

Ingeniería del caos

- Propuesto por Netflix en 2010 (*Chaos Monkey*)
- Introducir fallos adrede en los sistemas
- Probar sistemas distribuidos
  - Romper cosas a propósito
    - 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