Una aproximación ágil a las arquitecturas de servicios. Gestión distribuida de logs mediante contenedores Docker y Elastic Stack.

- Javier García Ros
- Jorge d'Ors Vilardebó

UNIVERSIDAD DE MURCIA

JORNADAS TÉCNICAS DE REDIRIS - Noviembre, 2016

ORIGEN DEL PROYECTO (I)

Oh my LOGS!!

GESTIÓN DE LOGS

- Sofware elegido de gestión de logs

Elastic Stack (anteriormente ELK)

ORIGEN DEL PROYECTO (II)

- Devops
- Gestión-de-configuración puppet
- Contenedores Docker
- Desarrollo Ágil
- Otros: Integración-Continua, Microservicios, etc.
- Oportunidad: incorporación de desarrollador al equipo

MANIFIESTO ÁGIL

Manifiesto Desarrollo Ágil

http://agilemanifesto.org/iso/es/manifesto.html

MANIFIESTO ARQUITECTURAS

Mi Manifiesto Desarrollo Ágil Arquitecturas :-)

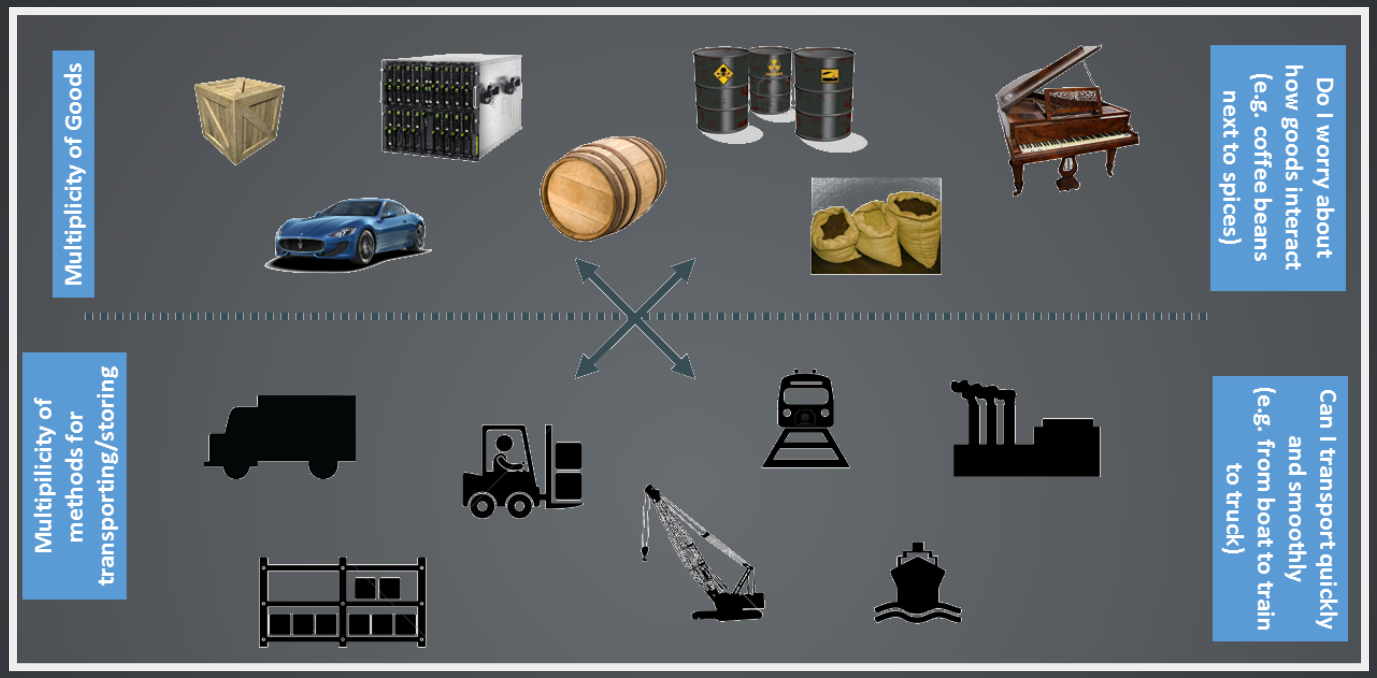
WHY?

“Code is Law”. Lawrence Lessig.

DOCKER: NEW KID...

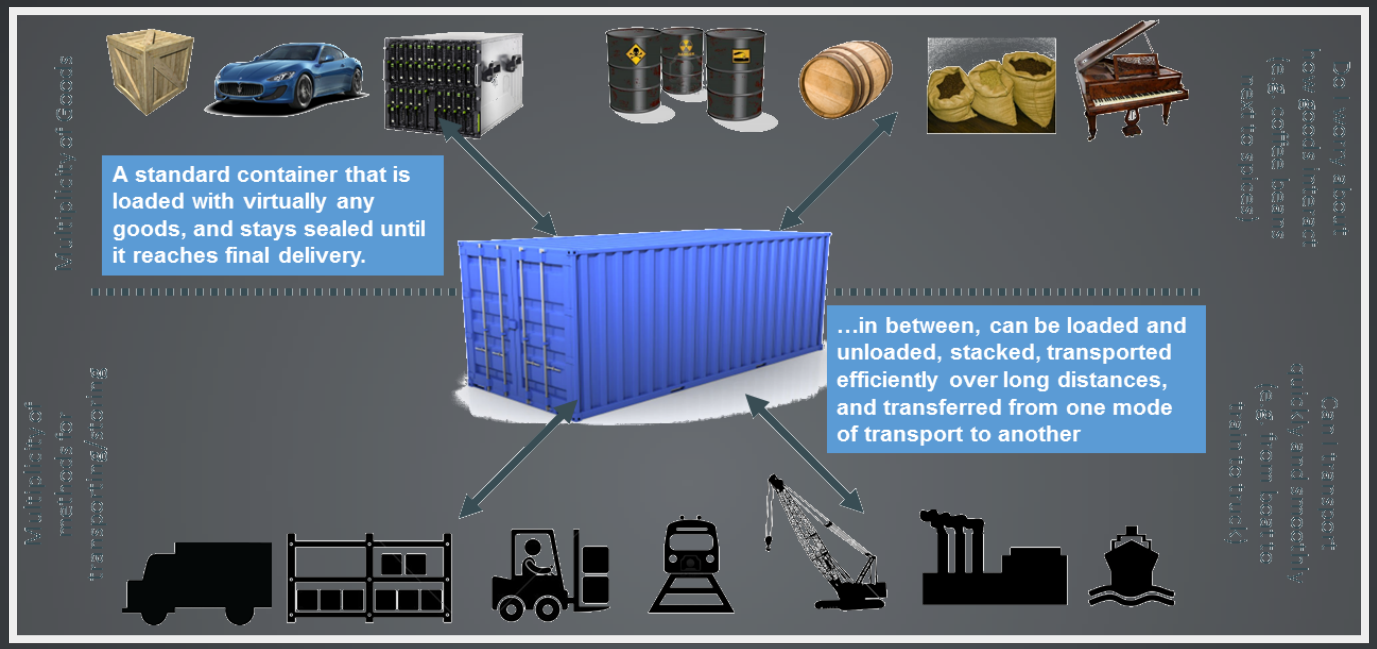
Contenedores

EL SIMIL



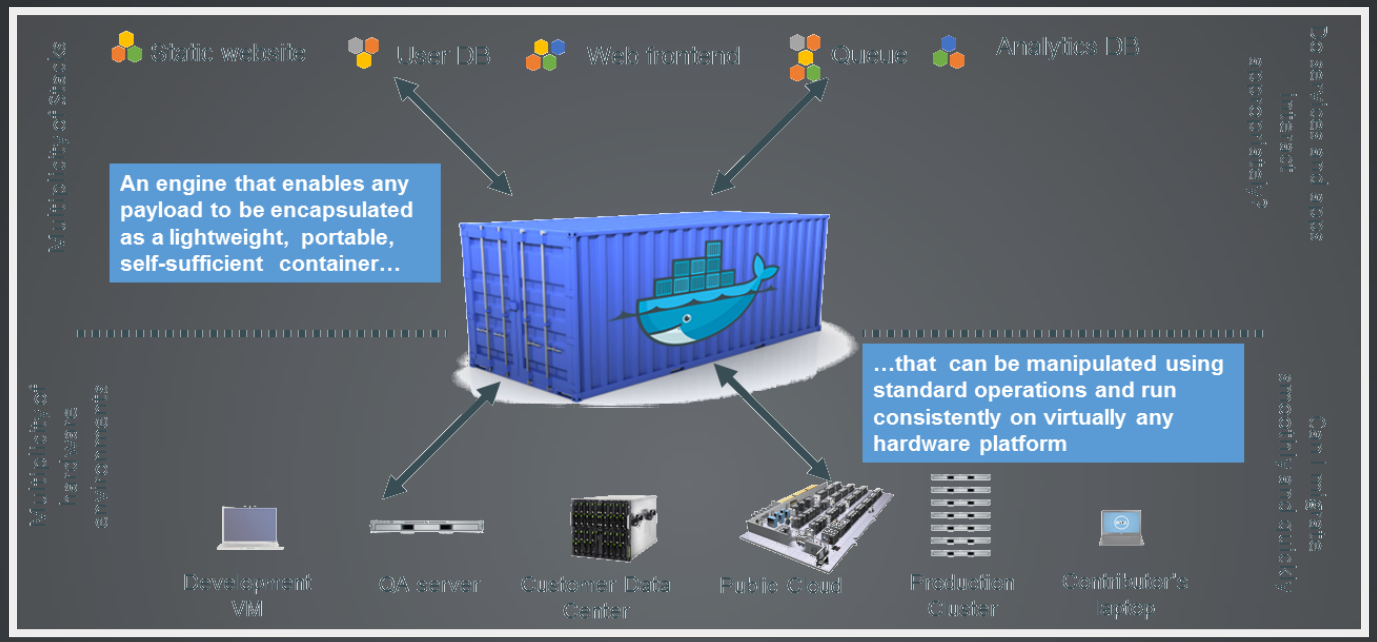
Problema del transporte antes-60

SOLUCIÓN



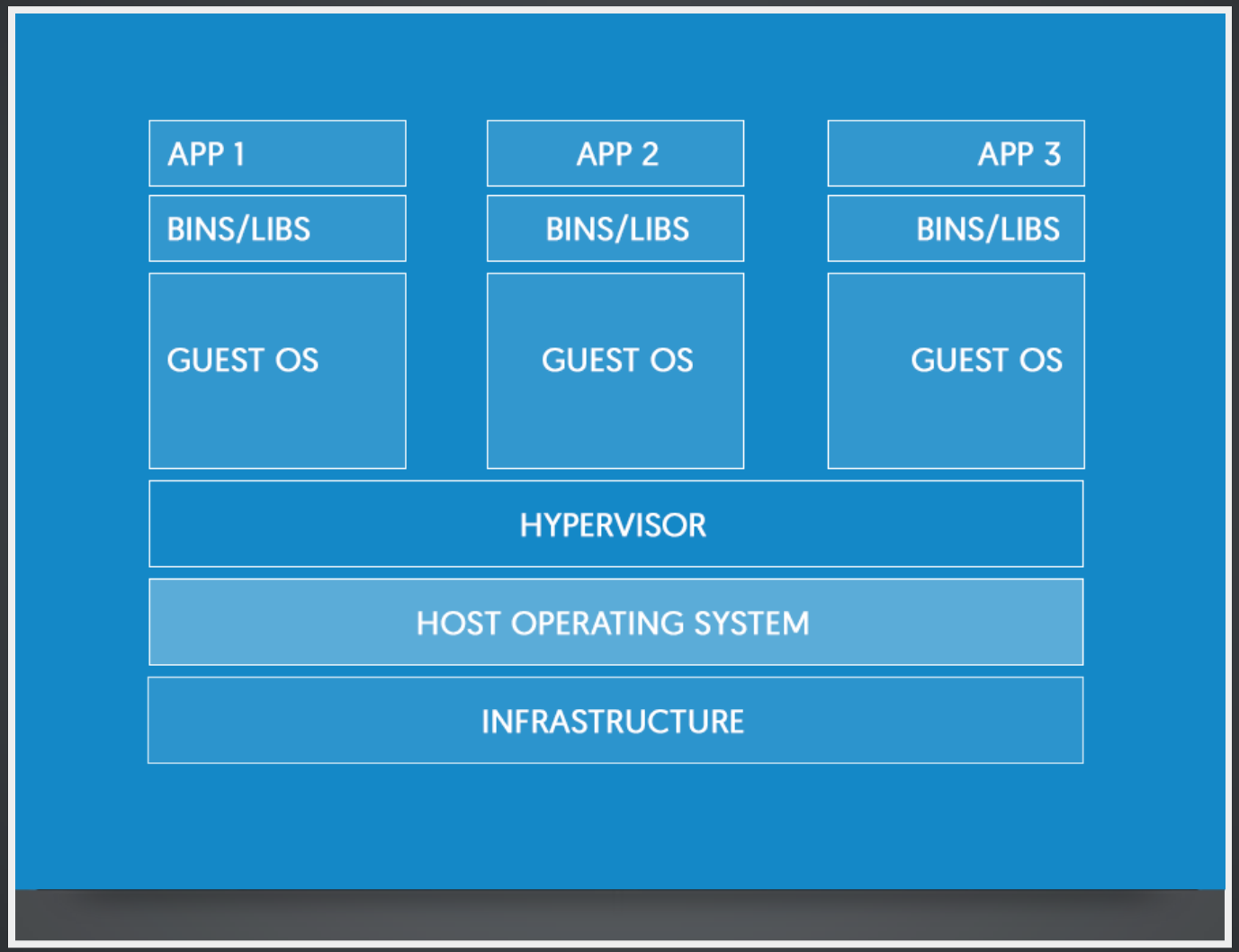
Contenedor intermodal

DOCKER: CONTENEDOR CÓDIGO



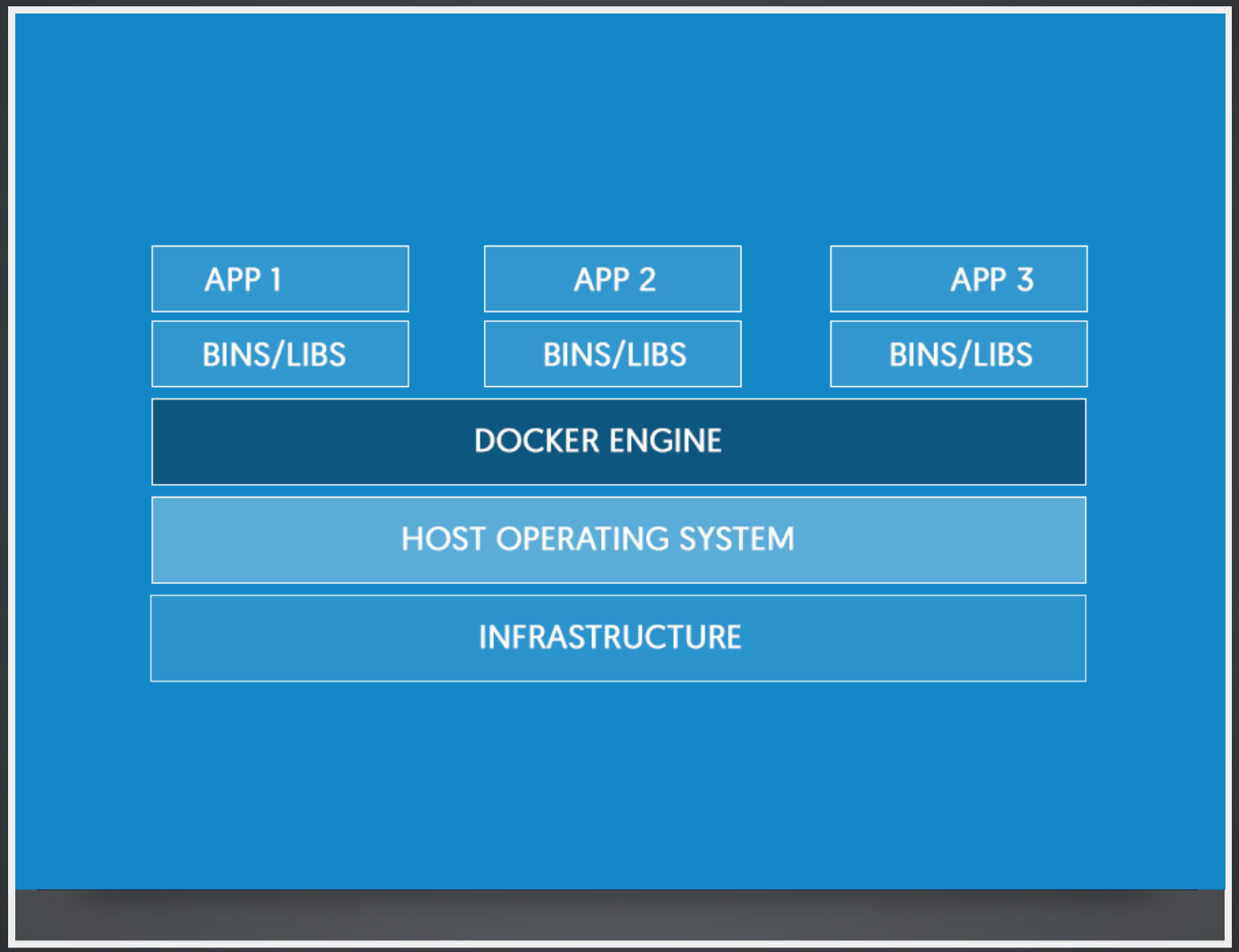
Portable, autocontenido, …

DOCKER VS VM (I)



Arquitectura de una VM

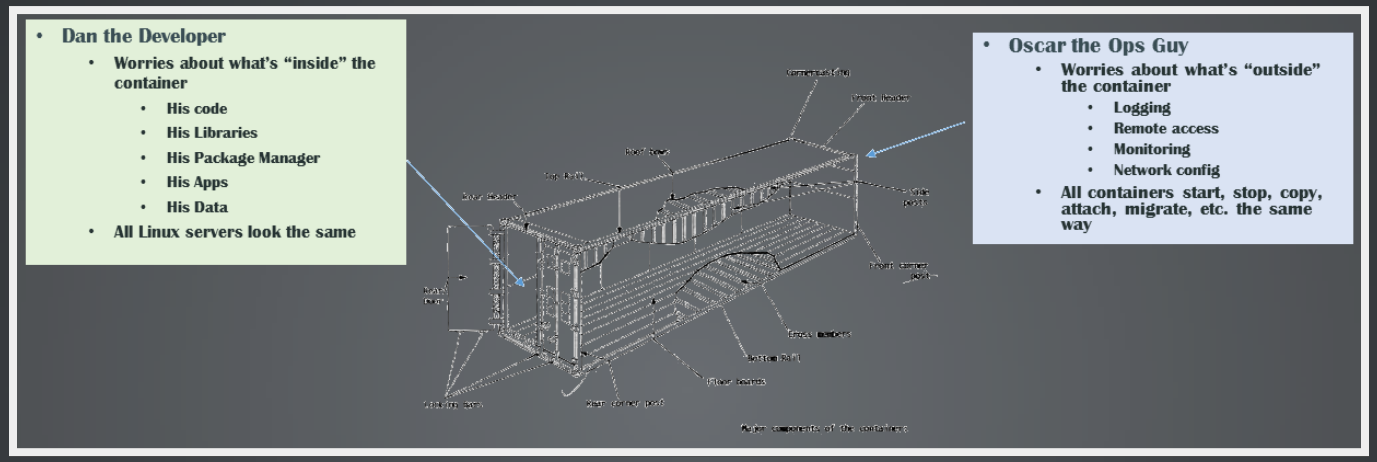
DOCKER VS VM (II)



Arquitectura Docker

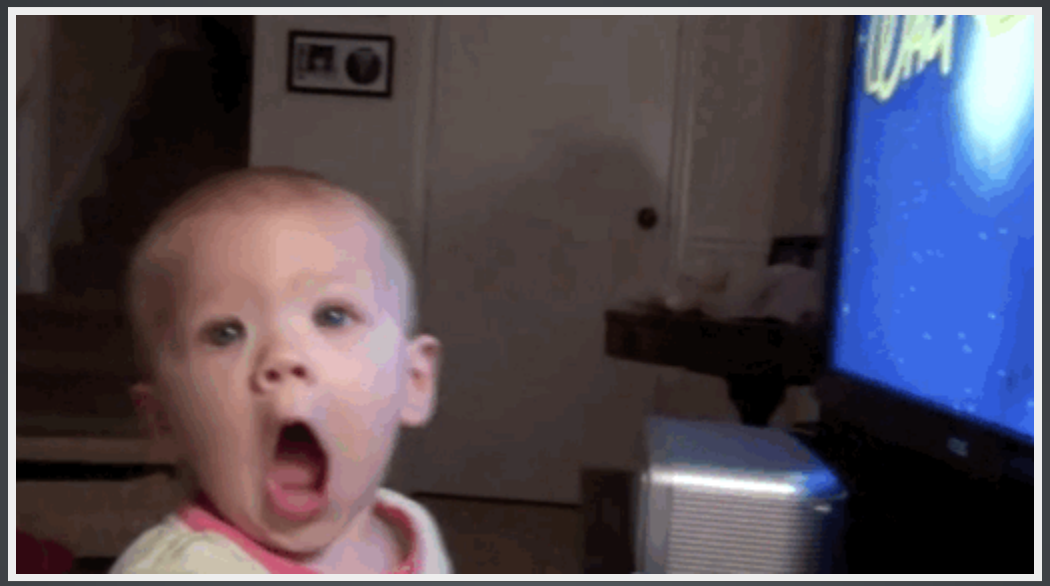
AMIGABLE DEVOPS

- Amigable para Desarrolladores y Administradores



Separación de intereses

FUNCIONA CON UN CLICK



A golpe de cl-ick (comand line :-)

```
docker run ubuntu echo Hello World
```

LAGAR

Recipiente donde se pisa la uva para obtener el mosto. ***RAE***

OBJETIVO FUNCIONAL

Facilitar el acceso y la explotación de la información

- Visualización
- Métricas
- Tendencias

OBJETIVOS TÉCNICOS

- Procesado de logs para el área TI
- Acceso independiente para aplicaciones/grupos
  - Autenticación con **SSO** (*CAS*)

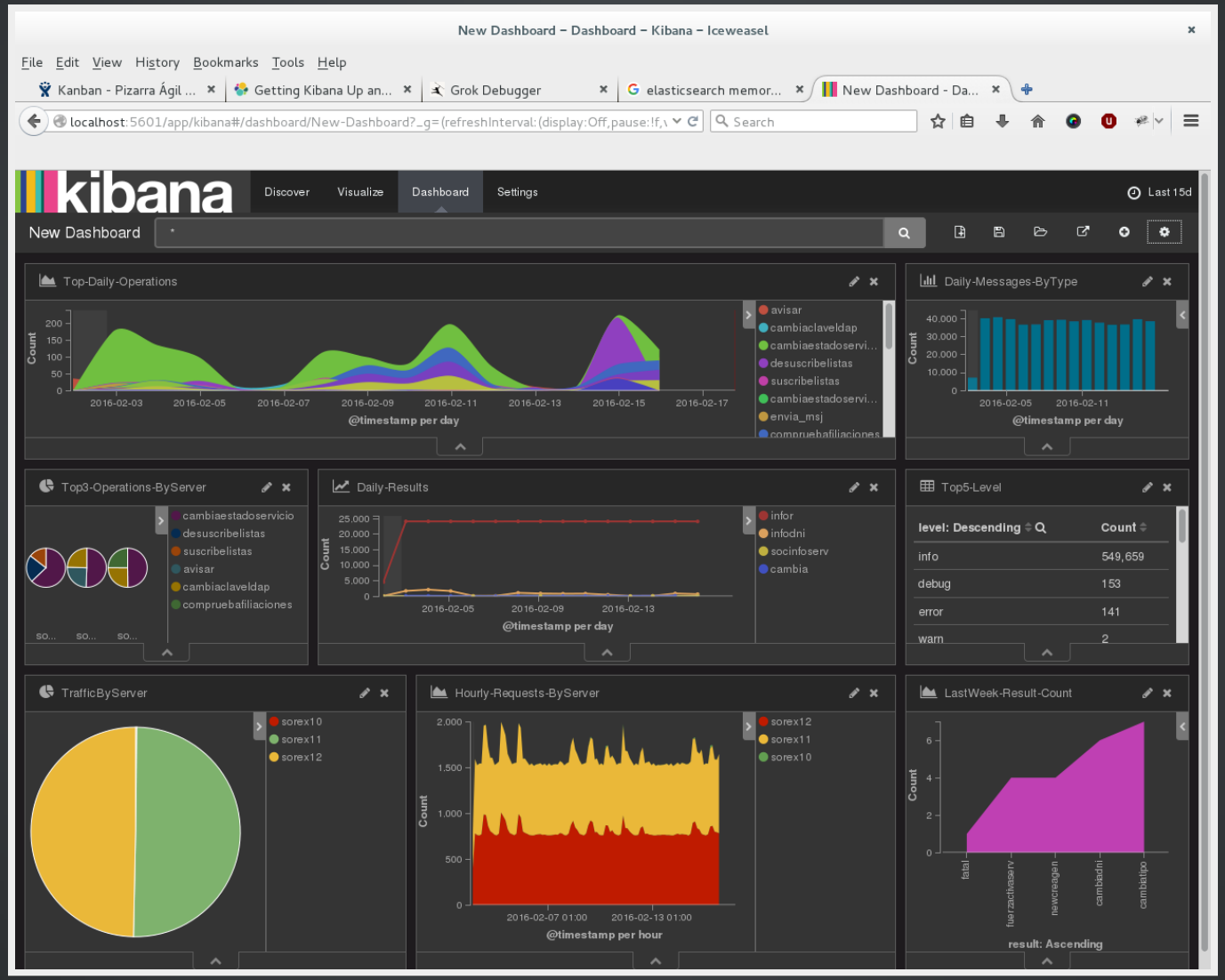
HERRAMIENTAS

- **ELK** como plataforma de gestión
- **Docker** para el despliegue
- **Puppet** para configuración de hosts
- **Redis** como tubería
- …

ELK STACK + REDIS

- **Elasticsearch**
  - Motor de consulta-noSQL
  - Basado en Lucene (índices)
- **Logstash**
  - Recolector y procesador de logs
- **Kibana**
  - Frontend Web
- **Redis**
  - BD en memoria (cola de mensajes)

ELK



Captura de pantalla de ejemplo

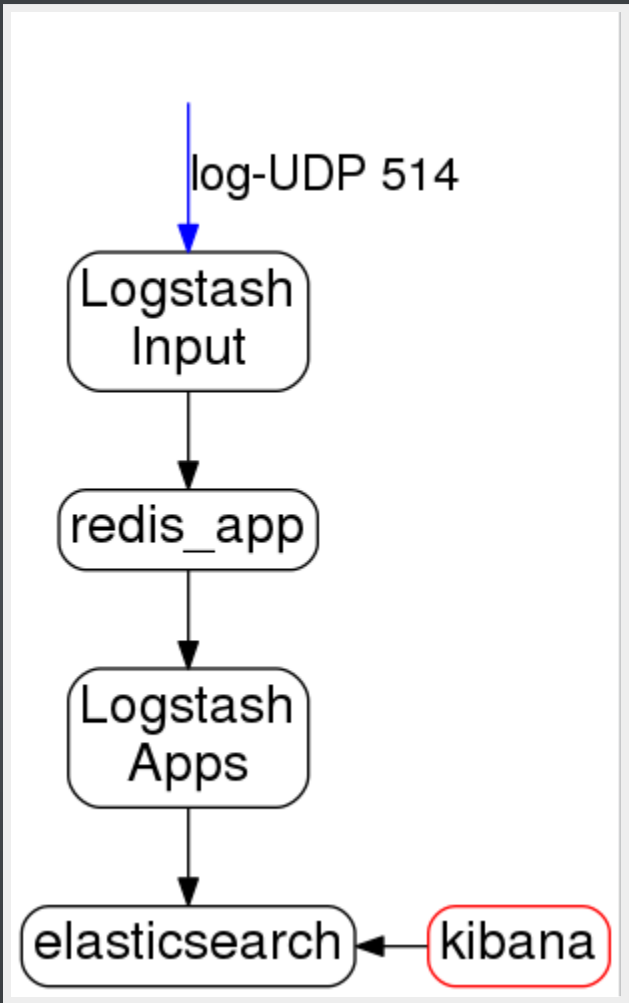
FASE I: OBJETIVOS

- Partiendo de un enfoque tradicional
- Toma de contacto con **Docker** y **ELK**
- Trabajo en local (**Infrastructure as Code**)
- Prueba básica de funcionamiento

FASE I: RESULTADOS

- La Prueba es satisfactoria
- Información útil desde el inicio
  - Detección de tráfico inesperado
  - Un servidor dentro de un cluster que no envía logs adecuadamente
  - Contenido de logs que no sigue *buenas prácticas*

FASE I: ARQUITECTURA



Arquitectura y flujo de logs

FASE I: EFECTOS SECUNDARIOS

- Necesidad de **RAM**
- Ocupación del disco: **100%**
  - Persistencia de contenedores
- *Grok nightmare*: importancia de normalizar los logs (**JSON**)

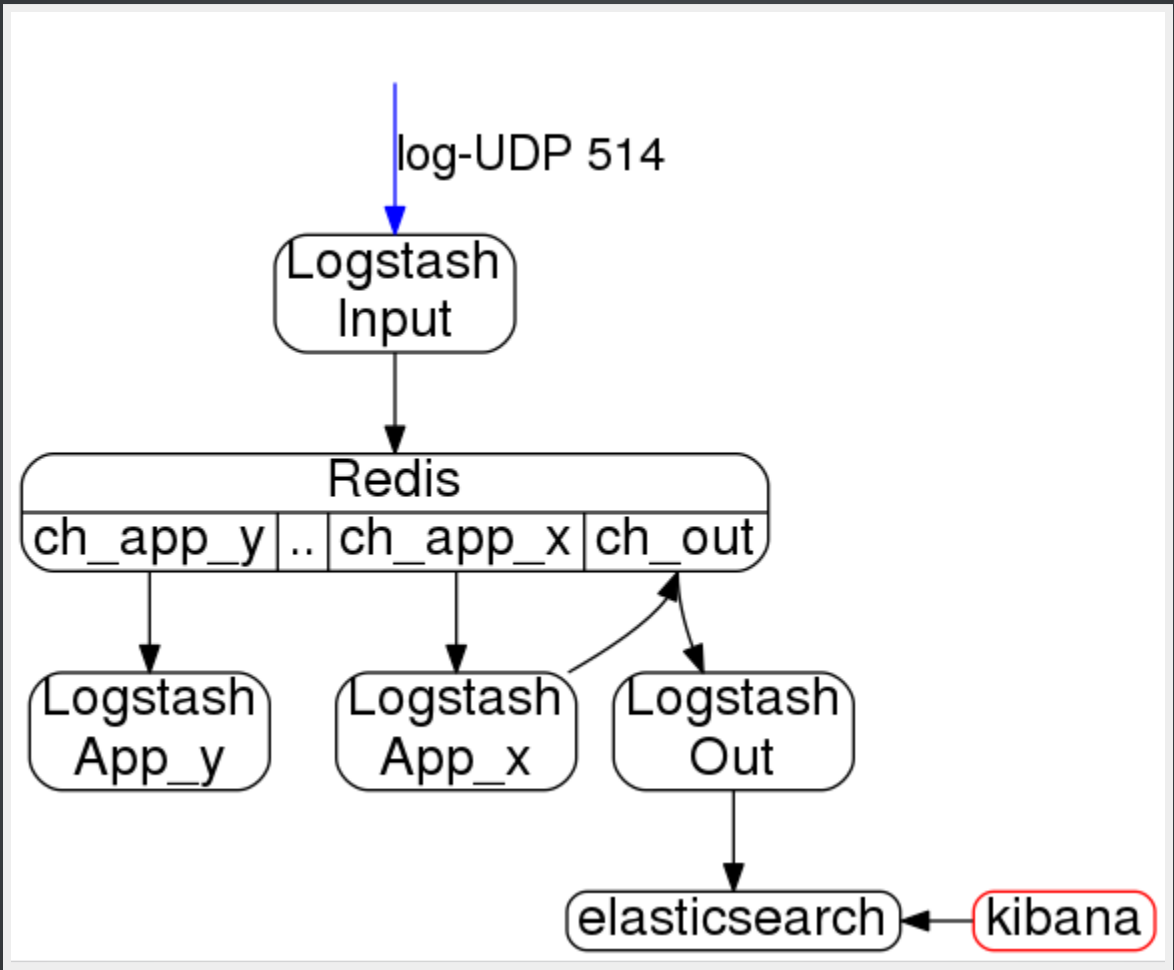
FASE II: OBJETIVOS

- Cambio de enfoque hacia una arquitectura flexible
  - *Múltiples contenedores*
- Servicios vs. Contenedores
  - *1 contenedor = 1 proceso*
- **Un contenedor por aplicación**
- **Redis**
- Control de la ocupación del disco (índices)

FASE II: RESULTADOS

- **Configuraciones independientes mucho más sencillas**
- Varias etapas de procesado
- *Basado en canales* **Redis**
  - es posible escuchar en 2º plano
- Procesado de 5 aplicaciones enviando logs
- Script de auto mantenimiento (basado en Curator)

FASE II: ARQUITECTURA



FASE II: EFECTOS SECUNDARIOS

FASE II: EFECTOS SECUNDARIOS

- La arquitectura se vuelve más compleja (OFM!)
- Pérdida de logs (UDP)
- Procesado de logs multilínea

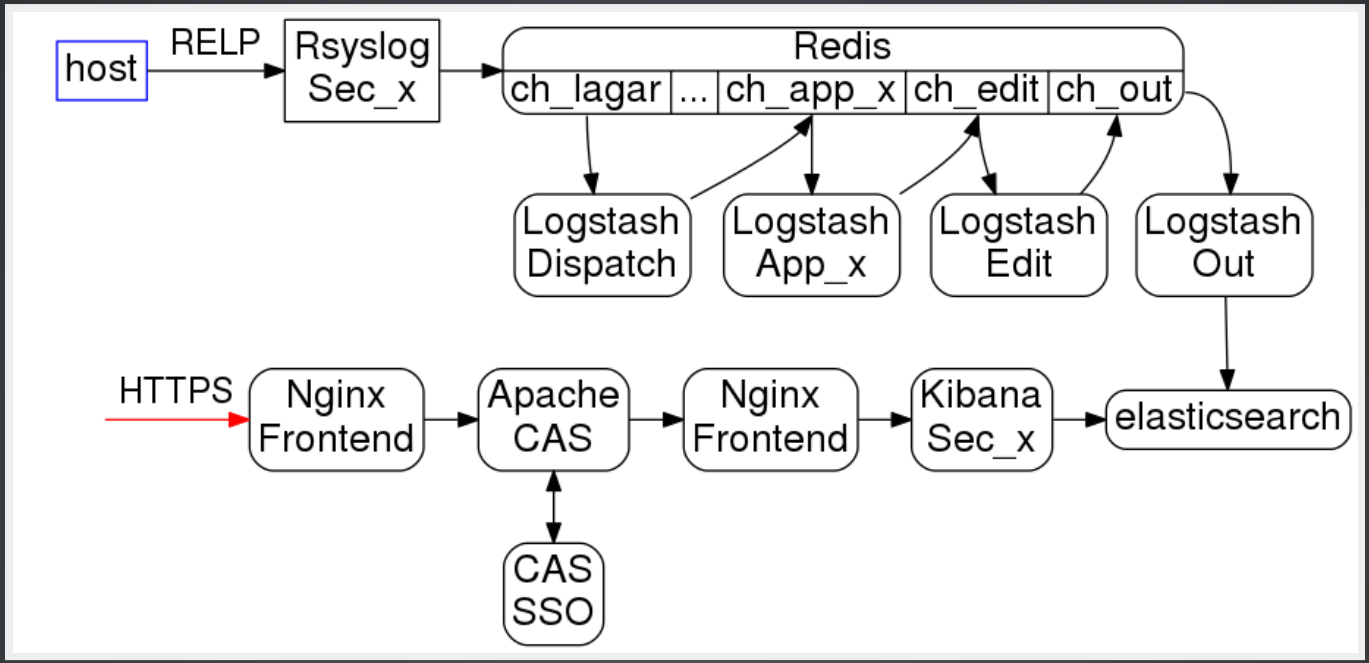
FASE III: OBJETIVOS

- Ausencia de AAA en **Kibana** (X-Pack)
  - **Autenticación y Autorización basada en SSO (CAS)**
- Contenedores por sección
- Transmisión confiable (**RELP**)
- Geoetiquetado
- Logs multilínea

FASE III: RESULTADOS

- Autenticación CAS:
  - Nginx-frontend + Apache2.4-CAS + Nginx-backend VHost
- Acceso independinte por sección
  - Contenedor Kibana por cada sección
- Procesado en origen: **rsyslog**.
  - *MaxMessageSize="32KB"*

FASE III: ARQUITECTURA



FASE III: EFECTOS SECUNDARIOS

- Do your grok grok?

> *grok (verb)* ***understand (something) intuitively or by empathy*****.**

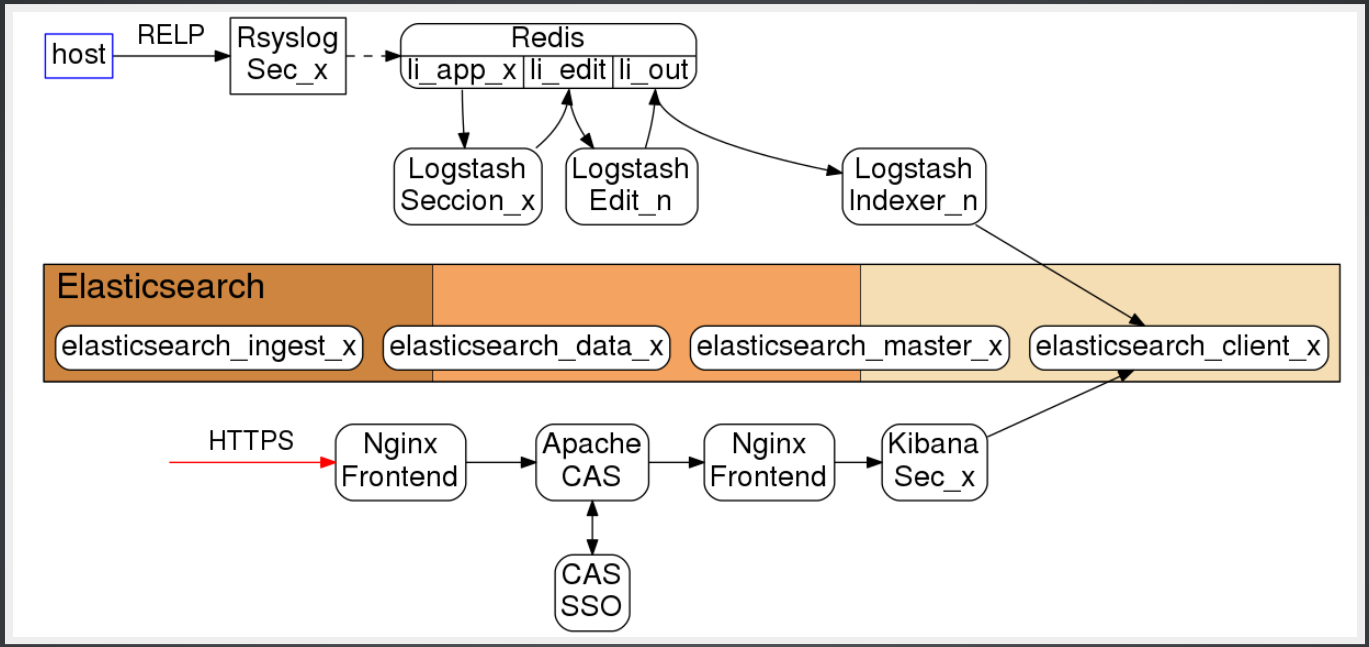
FASE IV: OBJETIVOS

- **Alta disponibilidad/cluster**
- Procesado de logs masivo: Aula Virtual/OFM
- Estudiar el comportamiento de la arquitectura
- Solucionar los problemas (github)

FASE IV: RESULTADOS

- Cluster **ELK**
- Cluster **Redis** vs. balanceo (múltiples Redis)
  - *Listas* Redis
- Ampliar recursos en etapas críticas
- **Github**: docker, elastic, redis, rsyslog

FASE IV: ARQUITECTURA



FASE IV: EFECTOS SECUNDARIOS

LECCIONES APRENDIDAS

- Aprendizaje
- Recursos hardware y humanos
- Normalizar vs. transformar la realidad
  - Cada aplicación *conoce* su formato
- Github

VÍAS FUTURAS

- Centralizar logs
- Archivado y firmado (ENS)
- Apache **Kafka** como alternativa a **Redis**
- Soluciones para la gestión de contenedores
  - *Docker swarm mode*
  - *Kubernetes*
- Monitorización
- Autogestión
- Mucho trabajo por hacer, mucho por mejorar

CONSIDERACIONES FINALES

- **IaC** (Infrastructure as Code)
- Architecture is abstract until operationalized. Neal Ford.
- automatización de infraestructuras
- “you build it, you run it!”"
- equipos según la cadena de valor (**Ley de Conway**)
- gran impacto en los SO de servidor

¿PREGUNTAS?

GRACIAS

“All genuine learning comes from experience” - John Dewey

- Javier García Ros
- Jorge d’Ors Vilardebó