



Contenedores de aplicaciones

M. en C. Gustavo Arellano



@arellano_gus



Introducción

- El objetivo de la conferencia, es mostrar una perspectiva objetiva y general de la tecnología de Contenerización, y su influencia en las arquitecturas de microservicios, vía la exposición y exploración de las características fundamentales de la tecnología y aspectos relacionados, como asincronía de servicios, orquestación de contenedores y herramientas de soporte a los procesos de "Continuous Integration", "Continuous deployment" y "Continuous Delivery".



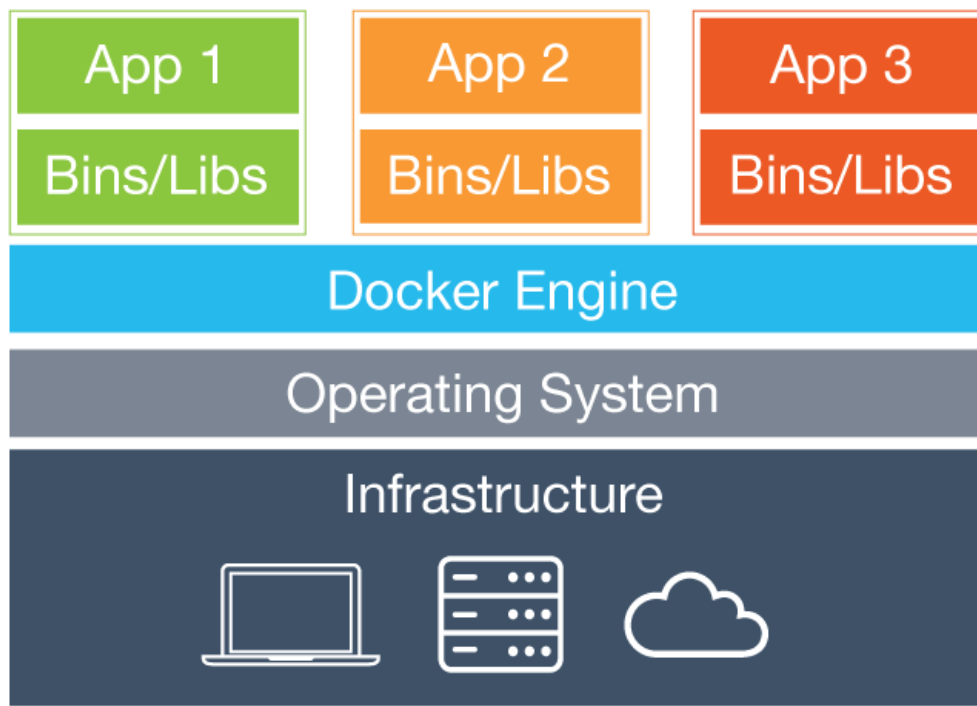
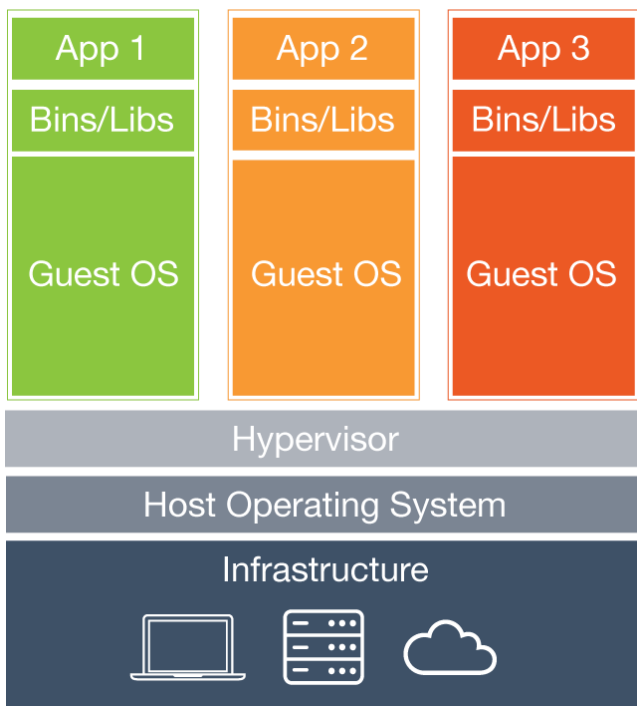
Definición 1

- **LXC (Linux Containers)** es una tecnología de virtualización a nivel de sistema operativo para Linux.
- LXC permite que un servidor físico ejecute múltiples instancias de sistemas operativos aislados, conocidos como Servidores Privados Virtuales (SPV o VPS en inglés) o Entornos Virtuales (EV).
- LXC no provee de una máquina virtual, más bien provee un entorno virtual que tiene su propio espacio de procesos y redes.



Contenedores versus Máquinas Virtuales

Los contenedores tienen algunas similitudes con las máquinas virtuales pero son diferentes en el enfoque arquitectónico, lo que hace a los contenedores mas portables y eficientes





Tipos de contenedores

- Como puede ser observado en: <https://coreos.com/rkt/docs/latest/rkt-vs-other-projects.html> existe una gran cantidad de propuestas que ofrecen mecanismos de contenerización:
 - Docker
 - RKT
 - runC
 - Containerd
 - OpenVZ
 - LXC/LXD
 - Systemd-nspawn
 - machineCtl
 - Qemu-kvm
- Sin embargo, durante la presente charla, nos concentraremos en los contenedores Docker



Docker

- Actualmente, Docker es una popular propuesta tecnológica que ofrece mecanismos de contenerización basado en tres características fundamentales:
 - cgroups
 - namespaces
 - FileSystem
- Los siguientes conceptos son fundamentales para el análisis de la propuesta Docker:
 - Host: La máquina en donde corren los Contenedores
 - Imagen: Binario que se usa como base para crear contenedores
 - Contenedor: Binario creado a partir de una imagen
 - Registro: Repositorio de imágenes
 - Volumen: Espacio externo a un contenedor
 - Dockerfile: Script usado para crear imágenes a partir de otras

Fuentes:

<https://blog.jayway.com/2015/03/21/a-not-very-short-introduction-to-docker/>

<https://www.slideshare.net/kerneltlv/namespaces-and-cgroups-the-basis-of-linux-containers>

<https://www.slideshare.net/jpetazzo/anatomy-of-a-container-namespaces-cgroups-some-file-system-magic-linuxcon>



Beneficios

(de los contenedores en general)

- Instalación y despliegue mas simple
- Independencia de plataforma
- Incremento en la eficiencia del proceso de virtualización
- Mejor y mayor aislamiento
- Mejor y mayor capacidad de gestión y automatización de contenedores

Fuente:

<https://www.1and1.mx/digitalguide/servidores/know-how/docker-container-las-ventajas-de-los-contenedores-web>



Impacto en DevOps

- La tecnología de contenerización contribuye con la relativamente reciente capa operativa denominada “devOps”
- Acelera y simplifica el proceso de despliegue de binarios en una variedad de ambientes de manera simultánea, eficiente, independiente y transparente
- Garantiza que los resultados locales sean consistentes con los obtenidos en despliegues remotos
- Realiza y hace concreto el proceso de despliegue continuo (continuous deployment) justo después de que se efectuó exitosamente el proceso de integración continua (continuous integration)



A jugar !!!

- Instalar docker:
 - `sudo apt-get update`
 - `wget -qO- https://get.docker.com/ | sh`
 - `sudo gpasswd -a ${USER} docker`
- Bajar una imagen:
 - `docker pull gustavoarellano/jdk18`
- Ejecutar una imagen:
 - `docker run -it gustavoarellano/jdk18 /bin/bash`
- Revisar estado general de contenedores:
 - `docker ps -a`
- Realizando un commit:
 - `docker commit c3f279d17e0a ejemplo:version1`
- Usando Dockerfile
 - `Docker build -t ejemplo .`



Parte II

Un ejemplo de uso práctico



Ciclo de vida del desarrollo de Software (con Docker)

01.- Control de Tareas

The screenshot shows a Jira board for a sprint named "KEB Sprint 5". The board is divided into columns: "To Do", "In Progress", "PR", and "Done".

- To Do:**
 - Leer acerca de ReactJS (KEB-139)
 - Preparar interfaces de Mobile (KEB-140)
 - Investigar regla 093 (Nomina12, KEB-168)
 - Crear rutina para validar de 1..N complementos de nomina (Nomina12, KEB-169)
 - Ajustar JIRA para que una actividad se pueda cancelar con estatus de "duoplicada" (KEB-178)
- In Progress:**
 - Crear módulo de generación de tickets haciendo uso de una base de datos. (KEB-211)
 - Crear Frontend para recibir y procesar ZIP's (incluye email, y # de ticket opcional) (KEB-210)
 - Documentar con javaDoc las clases de bajo nivel de la arquitectura (KEB-190)
 - Crear un script general que se encargue de realizar todo el proceso, desde descargar los modificadores método BuildList para (KEB-195)
- PR:** (Empty)
- Done:** (Empty)

On the right side, a detailed view of issue KEB-211 is shown:

- Issue:** KEB-211 (Kebblar / KEB-211)
- Description:** Crear módulo de generación de tickets haciendo uso de una base de datos.
- Estimate:** Unestimated
- Comments:** There are no comments yet on this issue.
- Attachments:** Drop files to attach, or browse.
- Sub-Tasks:** There are no sub-tasks.
- Development:** Create branch



Ciclo de vida del desarrollo de Software (con Docker)


02.- Control de Cambios

Create branch

Repository*

Branch from

Branch name*



develop

KEB-211-crear-modulo-de-generacion-de-ti



Ciclo de vida del desarrollo de Software (con Docker)

02.- Control de Cambios (b)

Java cfdi33

Kebblar Project / cfdi33

Commits

All branches

Find commits

Author	Commit	Message	Date	Builds
Ricardo Rodr...	d69bfeF	Actualizacion de xml, Reglas para soportar LCOQuery	2017-07-02	
garellano	98fe188	Se incorpora el plugin de "replace" de maven	2017-07-02	✓
garellano	29f22cb	ajustando el binary search	2017-07-02	✓
garellano	ad18cd5	Se agregan clases de soporte al módulo de validación	2017-07-02	✓
Ricardo Rodr...	3965221	Descarga de cambios y actualizacion LCO	2017-07-02	
Ricardo Rodr...	d15ef91	Actualizacion xml, Reglas 200	2017-07-02	
israel	77a7981	Actualizacion de clase LCO	2017-07-02	
israel	7193236	Actualizacion CFDI33Validator regla 017,018	2017-07-02	
israel	986897f	Actualizacion de regla 012	2017-07-01	
israel	b15c9a8	Se actualiza regla 012	2017-07-01	
garellano	7b9a379	ajustes minimos de formato	2017-07-01	⚠
garellano	1121b32	Se agrega el servicio de LCO y se actualiza su uso en las clases impactadas	2017-07-01	✓
Ricardo Rodr...	19a8811	Actualizacion y creacion xmIs,identificacion de xmIs a validar en reglas	2017-07-01	
israel	c4863a3	Se actualiza reglas 007, 025	2017-07-01	
israel	a8a26d7	Actualizacion de regla 05,09 y Archivos xml 200-00-007-01.xml 200-00-00...	2017-07-01	
israel	467eda8	Actualizacion de regla 05,09 y Archivos xml 200-00-007-01.xml 200-00-00...	2017-07-01	



Ciclo de vida del desarrollo de Software (con Docker)

02.- Control de Cambios (c)

The screenshot shows a GitHub pull request interface. On the left is a sidebar with navigation options like 'Clone', 'Create branch', 'Create pull request', 'Compare', 'Fork', 'Overview', 'Source', 'Commits', 'Branches', 'Pull requests', 'Pipelines', 'Downloads', 'Boards', and 'Settings'. The main content area shows the pull request details for 'Kebblar Project / cfdi33 / Pull requests'. The pull request is titled 'Agregando validación 3.3' and is currently 'OPEN'. It shows the author 'Francisco Gonzalez' and one reviewer. The description is 'No description'. Below the description, there are 'Comments (0)' and a text input field. The 'Files changed (2)' section lists two files: 'src/main/java/mx/qbits/kepler/validator/CFDI33Validator.java' with 6 additions and 1 deletion, and 'src/test/resources/testData/500-00-001-01.xml' with 115 additions and 0 deletions. The bottom part of the screenshot shows a diff view for the 'CFDI33Validator.java' file, with line numbers 84-89. The code shows a method 'rule001()' with a comment '// regla intencionalmente dejada en blanco' and two new lines of code: 'String keyVersion = "/cfdi:Comprobante[1]/@Version";' and 'String version = support.getStr(keyVersion);'. The diff highlights the changes in green and red.



Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua

Build projects
Kebblar

Project wallboard

Plan	Build	Completed	Tests	Reason	
01 cfdi-utils	✔ #17	2 weeks ago	No tests found	Manual run by Gustavo A Arellano S	▶ ✎ ☆
02 catalogos	✔ #42	1 week ago	No tests found	Changes by garellano <garellano@localhost>	▶ ✎ ☆
03 db-engine	✔ #11	1 week ago	No tests found	Changes by garellano <garellano@localhost>	▶ ✎ ☆
04 extractor	✔ #87	1 week ago	No tests found	Manual run by Gustavo A Arellano S	▶ ✎ ☆
05 cfdi	✔ #329	5 days ago	No tests found	Changes by garellano <garellano@localhost>	▶ ✎ ☆
06 engine-service (REST)	✔ #128	5 days ago	No tests found	Changes by garellano <garellano@localhost>	▶ ✎ ☆
currency-service (REST)	✔ #14	1 week ago	No tests found	Manual run by Gustavo A Arellano S	▶ ✎ ☆
keb-data	✔ #3	1 month ago	No tests found	Manual run by Gustavo A Arellano S	▶ ✎ ☆
lco-service (REST)	✔ #26	1 month ago	No tests found	Changes by Francisco Gonzalez <francisco.gonzalez@keblar.io>	▶ ✎ ☆
timbrado-service (REST)	✔ #4	2 months ago	No tests found	Changes by garellano <garellano@192.168.100.5>	▶ ✎ ☆



Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua (b)

Build projects / Kebblar / 06 engine-service (REST)
Configuration - 06 engine-service (REST)

Motor Orquestador

Plan Configuration

- Stages & jobs 1
 - Default Stage
 - Default Job**
 - Branches 0

Job details | Tasks | Requirements | Artifacts | Miscellaneous

Tasks

A task is a piece of work that is being executed as part of the build. The execution of a script, a shell command, an Ant Task or a Maven goal are only few examples of Tasks. Learn more about tasks.

You can use [runtime](#), [plan](#) and [global variables](#) to parameterize your tasks.

1 agent has the [capabilities](#) to run this job

Source Code Checkout

Checkout Default Repository

Command

```
rm -rf /var/www/intranet/site/kebbllar/engine
```

Command

```
mkdir -p /var/www/intranet/site/kebbllar/engine
```

Command

```
mvn -U clean install site deploy
```

Command

```
mvn javadoc:javadoc -Dadditionalparam=-Xdoclint:none
```

Command

```
cp -R target /var/www/intranet/site/kebbllar/engine
```

Command

```
rm -rf /home/ubuntu/.m2/repository/mx/qbi /root-kebbll*
```

SCP Task

```
scp -i kebbllar-ambientes-cer engine-0.0.1-SNAPSHOT-fat.jar ubuntu@52.44.208.181:/home/ubuntu/binaries/dev
```

SSH Task

SCP Task configuration

Task description

```
scp -i kebbllar-ambientes-cer engine-0.0.1-SNAPSHOT-fat.jar ubuntu@52.44.2
```

Disable this task

Host*

Hostname or IP address of the remote host

Username*

Username you want to use to access the remote host

Authentication Type*

Change SSH Key

Artifact

Local Path*



Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua (c)

Browse

- Welcome
- + Search
- Browse
- Assets
- Components**

Components / kebbлар-group

Filter

	Group ↑	Name	Version	
	mx.qbits.kepler	catalogos	0.01-20170726.180844-7	>
	mx.qbits.kepler	catalogos	0.01-20170728.032842-8	>
	mx.qbits.kepler	catalogos	0.01-20170730.183845-9	>
	mx.qbits.kepler	catalogos	0.01-20170807.015943-10	>
	mx.qbits.kepler	cfdi33	0.01-20170728.002324-74	>
	mx.qbits.kepler	cfdi33	0.01-20170728.030223-75	>
	mx.qbits.kepler	cfdi33	0.01-20170728.031425-76	>
	mx.qbits.kepler	cfdi33	0.01-20170728.031725-77	>
	mx.qbits.kepler	cfdi33	0.01-20170728.032954-78	>
	mx.qbits.kepler	cfdi33	0.01-20170728.041425-79	>
	mx.qbits.kepler	cfdi33	0.01-20170728.155925-80	>
	mx.qbits.kepler	cfdi33	0.01-20170728.160226-81	>
	mx.qbits.kepler	cfdi33	0.01-20170728.04425-82	>



Ciclo de vida del desarrollo de Software (con Docker)

04.- Continuous Deployment

Stack: Default Add Service Active

Active	currency-service-dev	Image: keblar-dock:latest	Service	1 Container
Active	currency-service-lb-dev	To: currency-service-dev Ports: 11000/tcp	Load Balancer	1 Container
Active	docker-nginx-dev	Image: nginx	Service	1 Container
Active	docker-nginx-lb-dev	To: docker-nginx-dev Ports: 14000/tcp	Load Balancer	1 Container
Active	endpoint-dev	Image: keblar/cfdi33	Service	2 Containers
Active	endpoint-lb-dev	To: endpoint-dev Ports: 80/tcp	Load Balancer	1 Container
Active	lco-dev	Image: keblar-dock:latest	Service	2 Containers
Active	lco-lb-dev	To: lco-dev Ports: 10000/tcp	Load Balancer	1 Container
Active	prueba-ssl	To: lco-dev Ports: 443/tcp	Load Balancer	1 Container
Active	timbrado-dev	Image: keblar-dock:latest	Service	1 Container
Active	timbrado-service-lb-dev	To: timbrado-dev Ports: 13000/tcp	Load Balancer	1 Container
Active	z-co-downloader	Image: download-cron	Service	1 Container



Ciclo de vida del desarrollo de Software (con Docker)

04.- Continuous Deployment (b)

Service: endpoint-dev in Default Active

Description:
Servicio de validacion de CFDI 3.3

Type:
Service

Scale:
2

Image:
keblar/cfdi33

Entrypoint:
None

Command:
java -jar /binaries/dev/engine-0.0.1-SNAPSHOT-fat.jar

Ports Containers Labels Log

State	Name	IP Address	Host	Image	Stats
Running	Default-endpoi...	10.42.212.185	ip-172-31-6-3...	keblar/cfdi33	
Running	Default-endpoi...	10.42.120.5	ip-172-31-6-3...	keblar/cfdi33	



Ciclo de vida del desarrollo de Software (con Docker)

04.- Continuous Deployment (c)

```
JSON  Datos sin procesar  Cabeceras
Guardar Copiar
queryResult:
  0:
    rfc: "MET020809R70"
    certificado: "00001000000401230506"
    fechaInicial: "2016-01-21T03:33:52"
    fechaFinal: "2020-01-21T03:33:52"
  1:
    rfc: "MET020809R70"
    certificado: "00001000000305287814"
    fechaInicial: "2014-10-12T02:22:16"
    fechaFinal: "2018-10-12T02:22:16"
info:
  totalNumRegs: "13,780,477"
  Current Node IP: "10.42.191.2"
  totalResults: "2"
  Call Timestamp: "Mon Aug 14 23:52:37 UTC 2017"
  timeToProcess: "0.450051 milisegundos"
  Caller IP: "10.42.231.35"
```



Gracias !

G. Arellano
@arellano_gus
2017