Contenedores de aplicaciones

M. en C. Gustavo Arellano

 @arellano_gus

Introducción

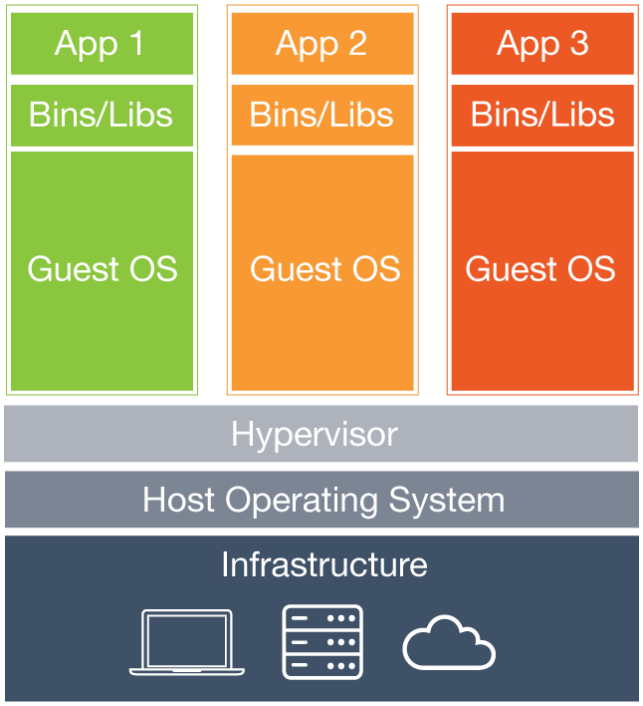
- El objetivo de la conferencia, es mostrar una perspectiva objetiva y general de la tecnología de Contenerización, y su influencia en las arquitecturas de microservicios, vía la exposición y exploración de las características fundamentales de la tecnología y aspectos relacionados, como asincronía de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

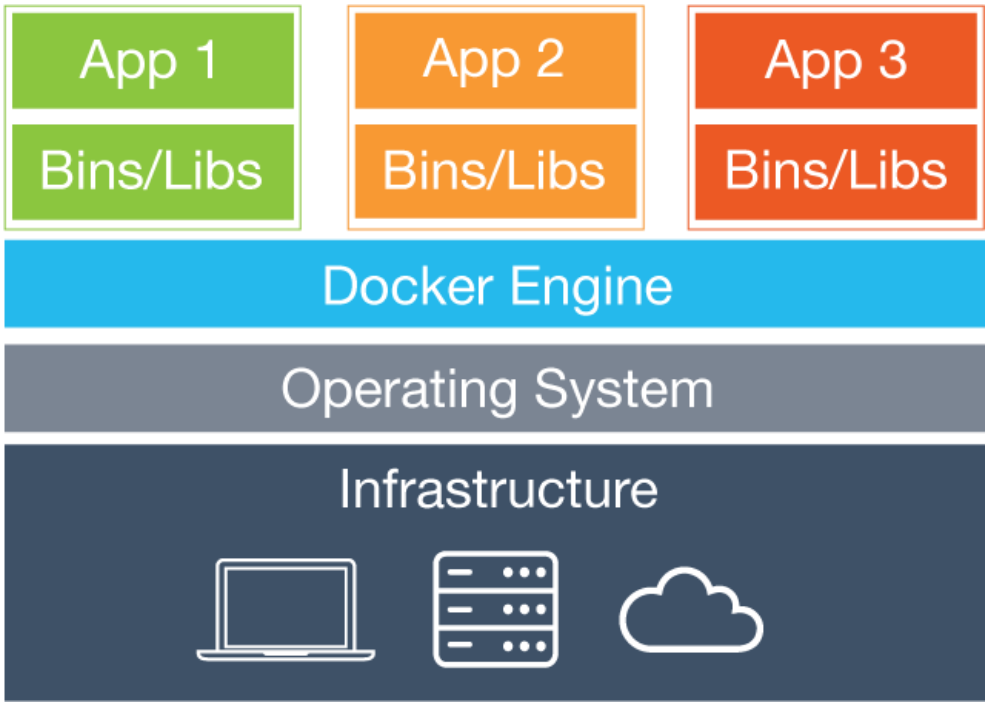
Definición 1

- **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-filesystem-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 !!!

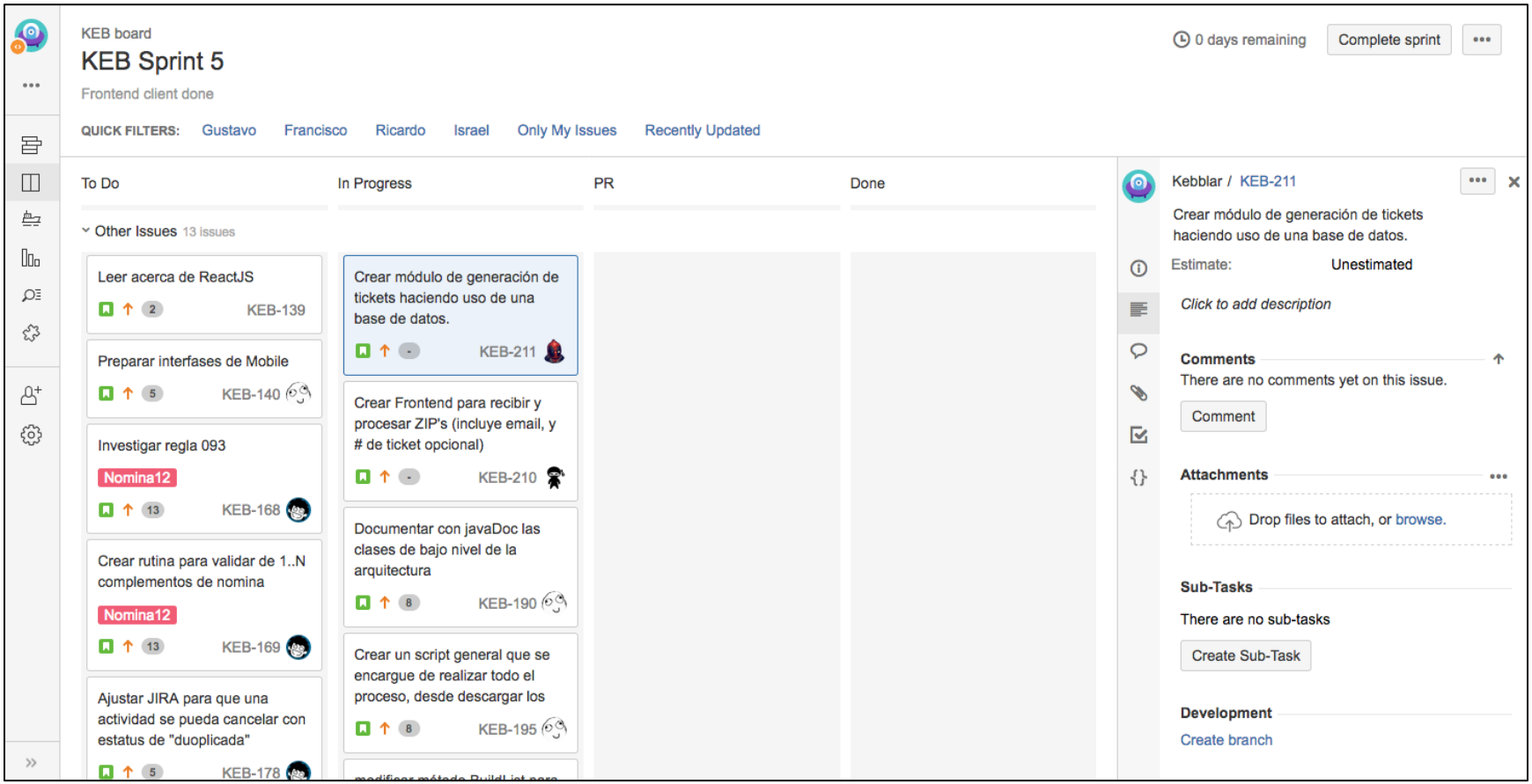
- Instalar docker:
  - sudo apt-get update
  - wget -qO- https://get.docker.com/ | sh
  - sudo gpasswd -a ${USER} docker
- Bajar una imagen:
  - docker pull gustavoarellano/jdk18
- Ejecutar una imagen:
  - docker run –it gustavoarellano/jdk18 /bin/bash
- Revisar estado general de contenedores:
  - docker 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

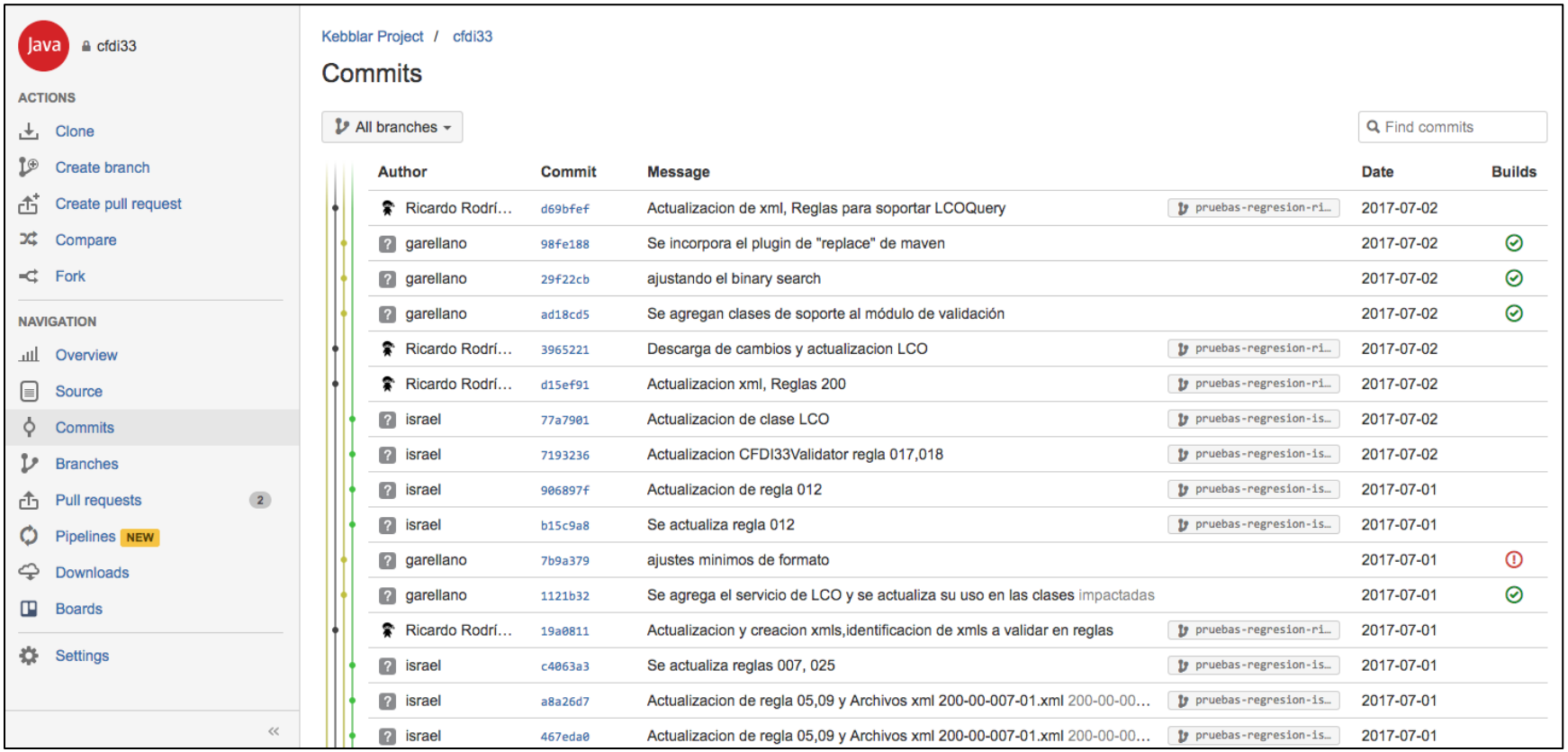


Ciclo de vida del desarrollo de Software (con Docker)

02.- Control de Cambios

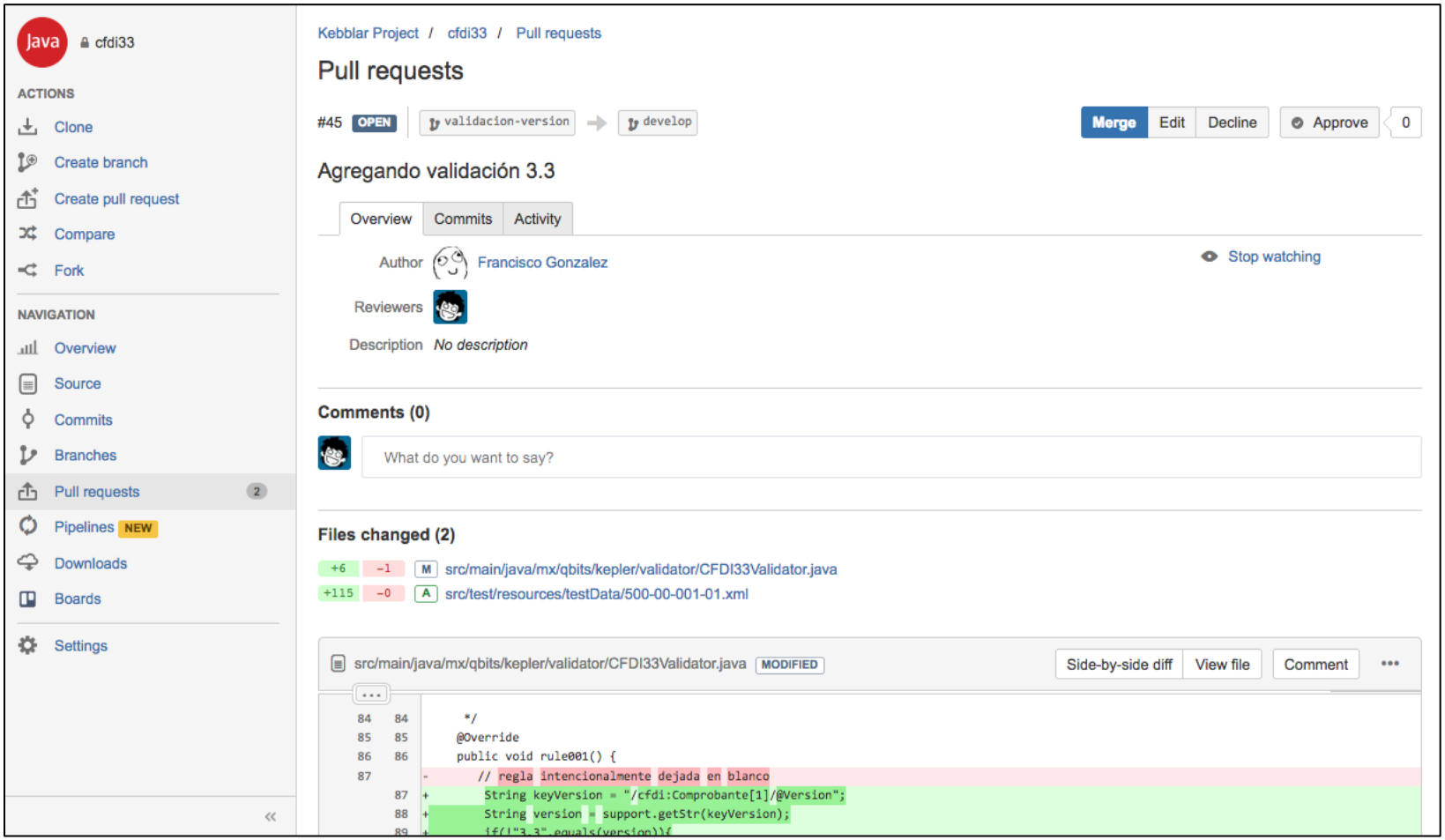
Ciclo de vida del desarrollo de Software (con Docker)

02.- Control de Cambios (b)



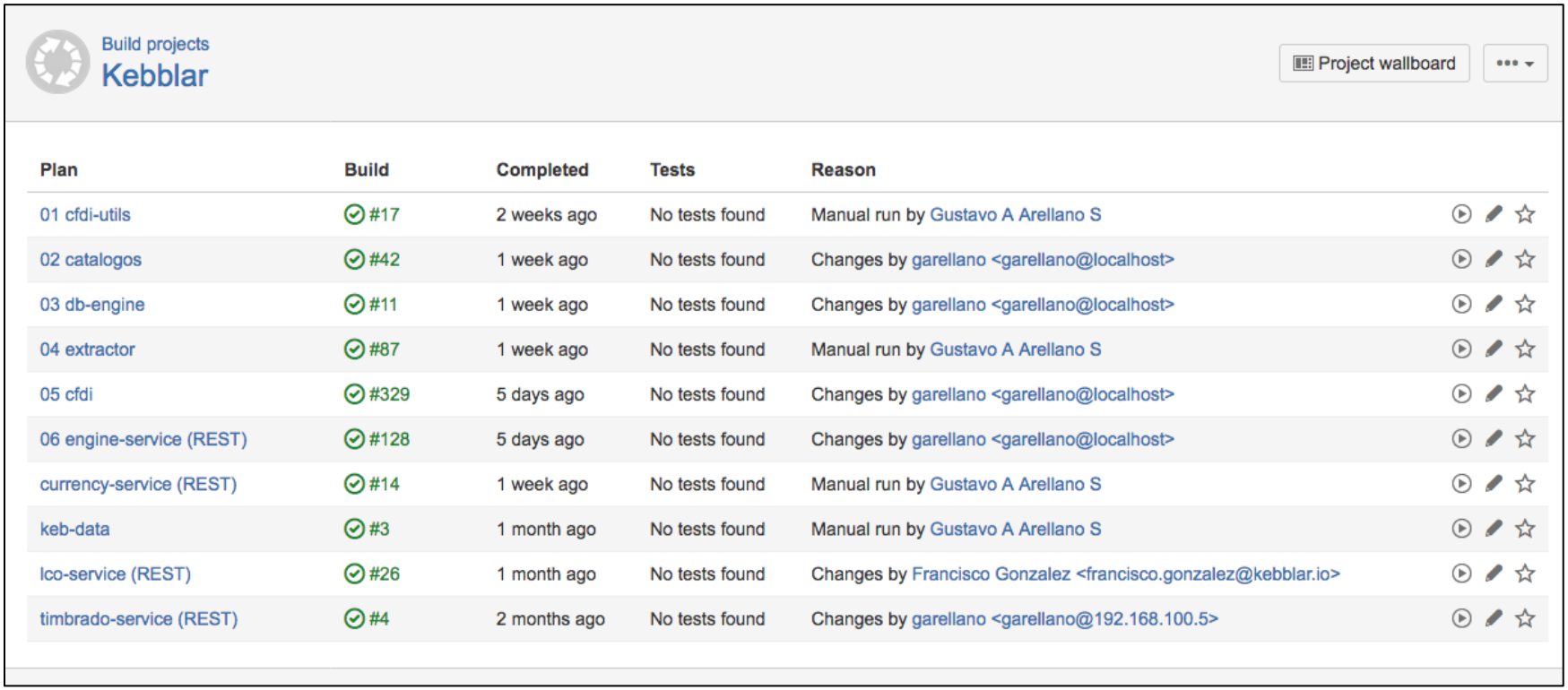
Ciclo de vida del desarrollo de Software (con Docker)

02.- Control de Cambios (c)



Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua



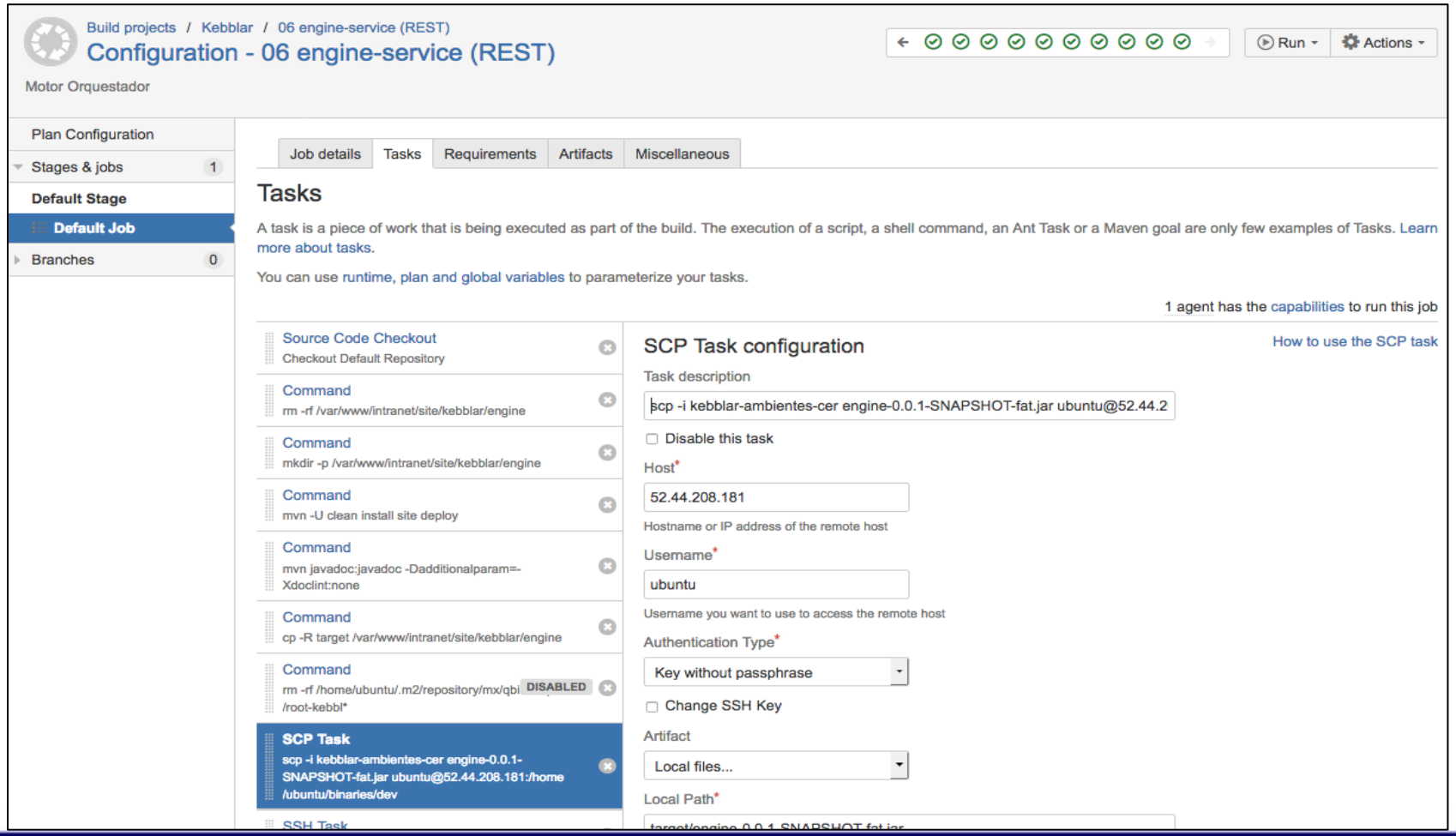
Build projects
Kebblar

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@kebblar.io>
timbrado-service (REST)	#4	2 months ago	No tests found	Changes by garellano <garellano@192.168.100.5>

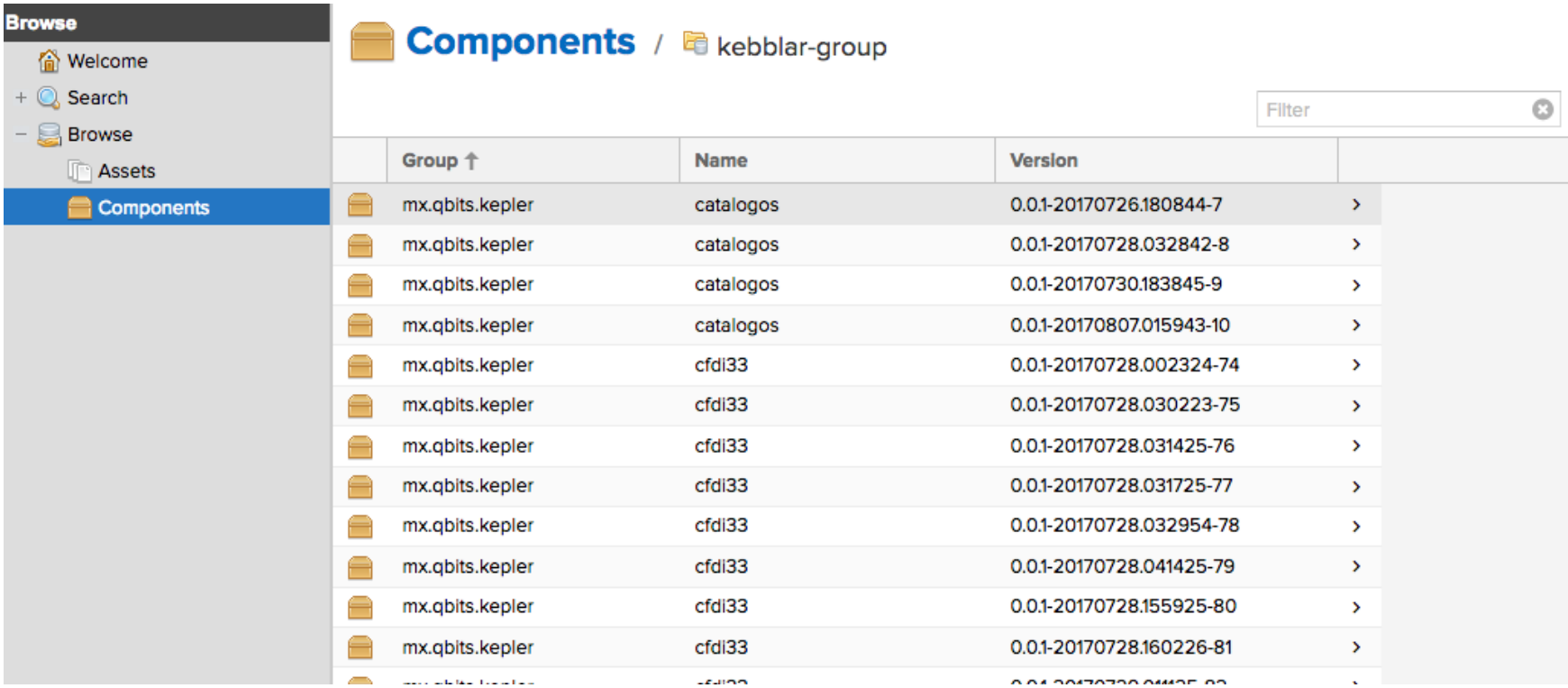
Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua (b)



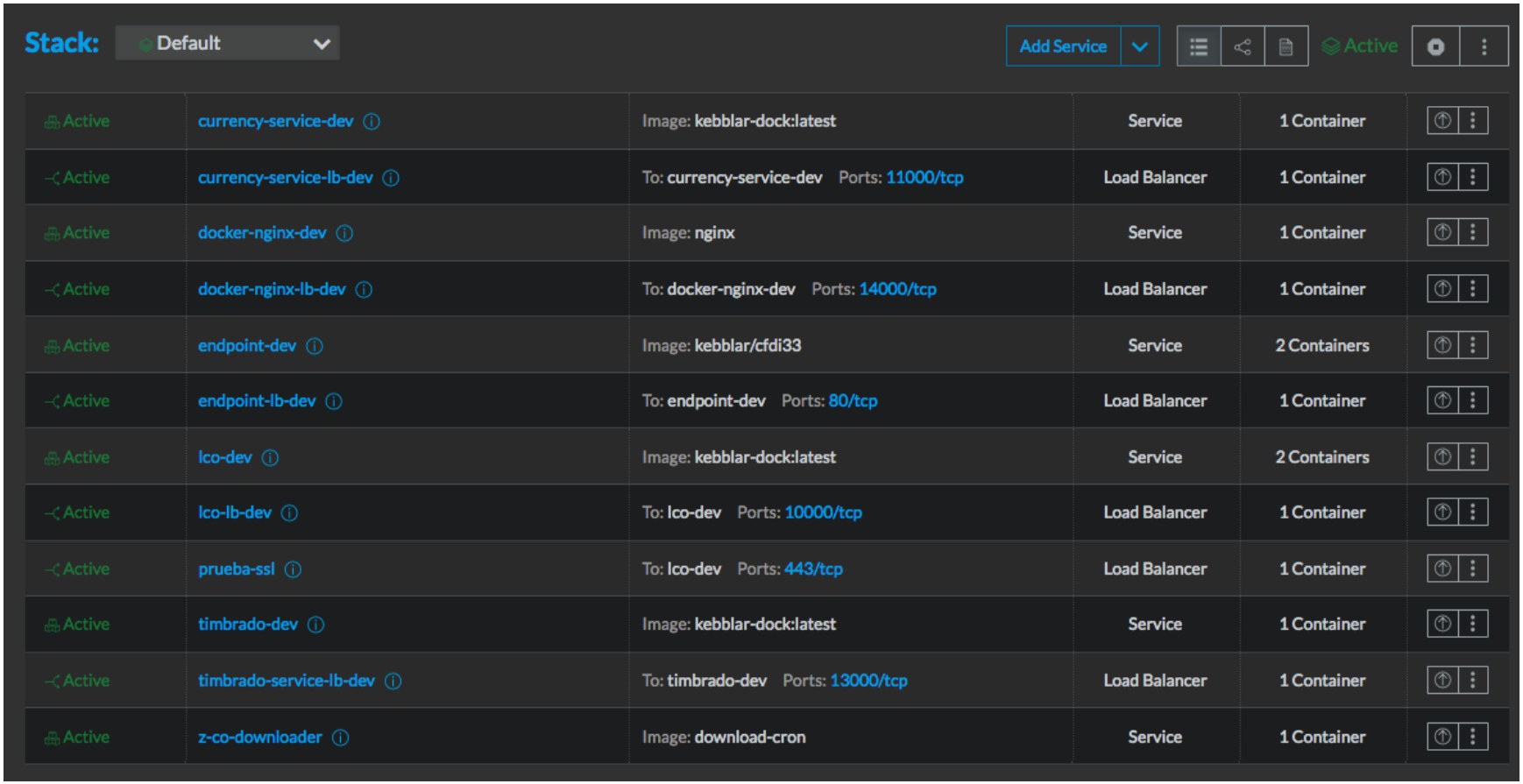
Ciclo de vida del desarrollo de Software (con Docker)

03.- Integración Continua (c)



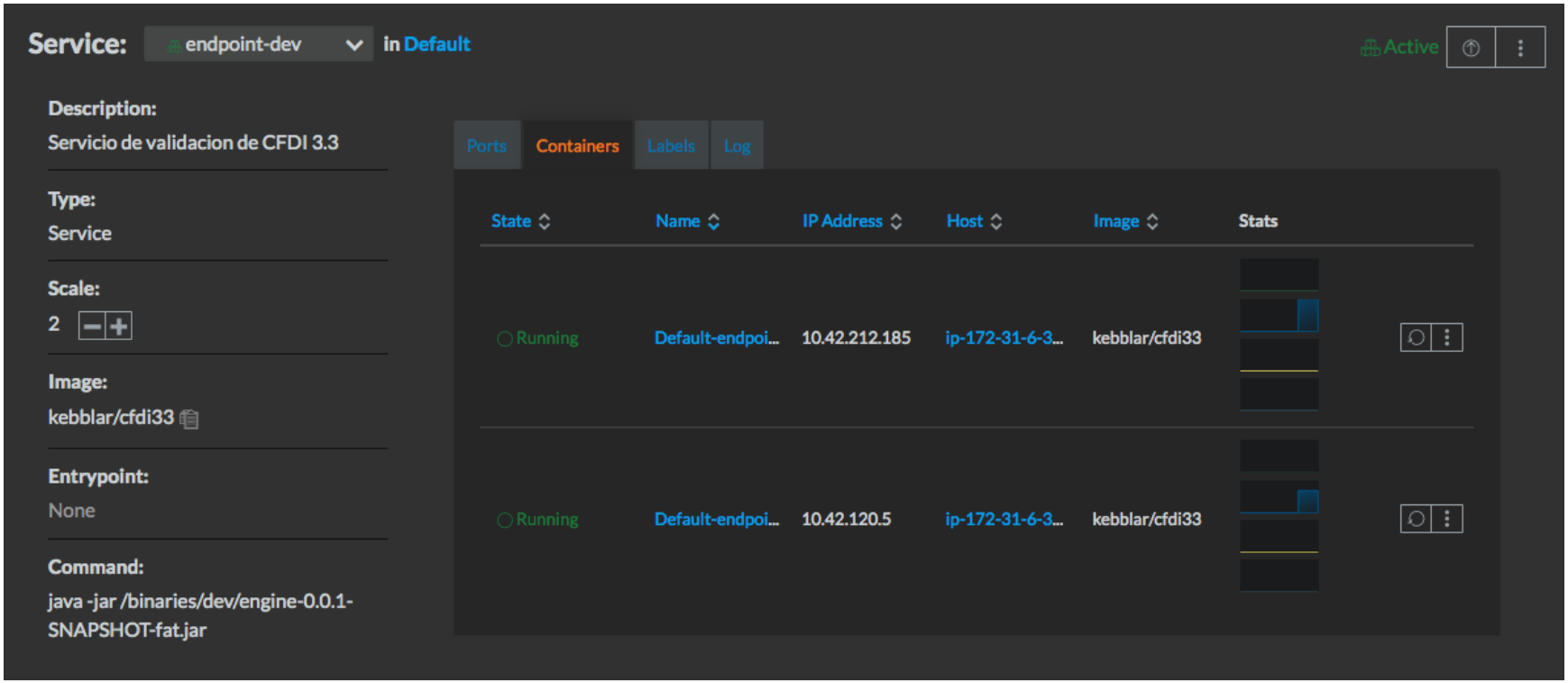
Ciclo de vida del desarrollo de Software (con Docker)

04.- Continuous Deployment



Ciclo de vida del desarrollo de Software (con Docker)

04.- Continuous Deployment (b)



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