DOXMX — 31/10/2019

#CI/CD #rkt #Microservices #Containerd #Swarm #Rancher #SOA #k8s #Docker #API #Mesos #Python #Docker Compose #60

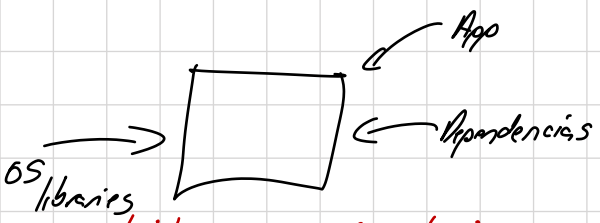
Contenedores:

Antecedentes:

- Retos en desarrollo y distribución de software.
  a) En mi maquina si funcionaba
  b) Tiempos para agregar nuevas funcionalidades
  c) Aprovechamiento de recursos
  d) Cattle VS Pets (No a las mascotas!!)

Contenedor:

- Recipiente inmutable que contiene los elementos estrictamente necesarios para la ejecución de un servicio o aplicación.



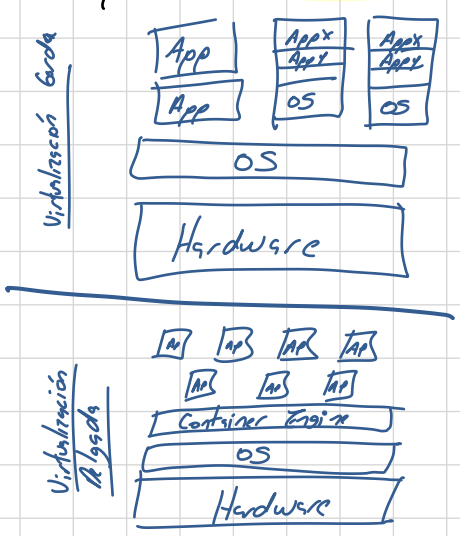
* Portable * Repetible * Inmutables
* Aislado * Remplazables
(Persistencia)

No es algo nuevo:

* LXC * Solaris Containers
* Chroot * OpenVZ

- Hace uso de las caracteristicas del kernel de Linux:
  a) cgroups - Iptables (Netfilter)
  b) namespaces

Tipo de virtualización:

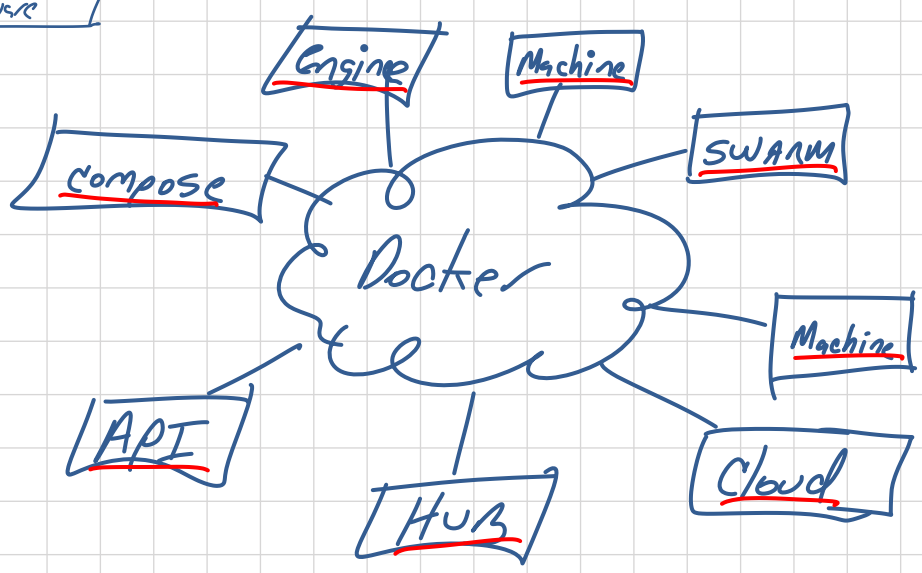


Elementos básicos:

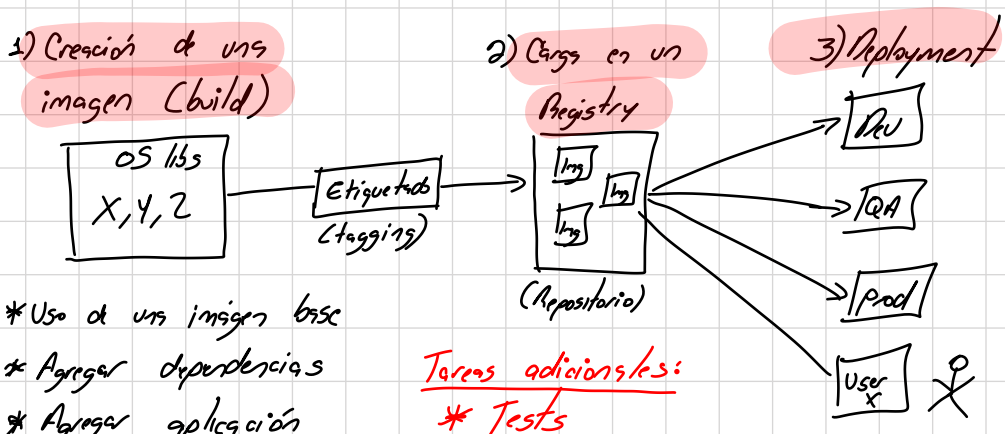
a) Imágen (Molde: SO+SW+config)
b) Contenedor (Instancia del Molde)
c) Engine (Crea el ambiente de ejecución para los contenedores)

Implementaciones actuales:

* rkt * coreOS * docker
* containerD



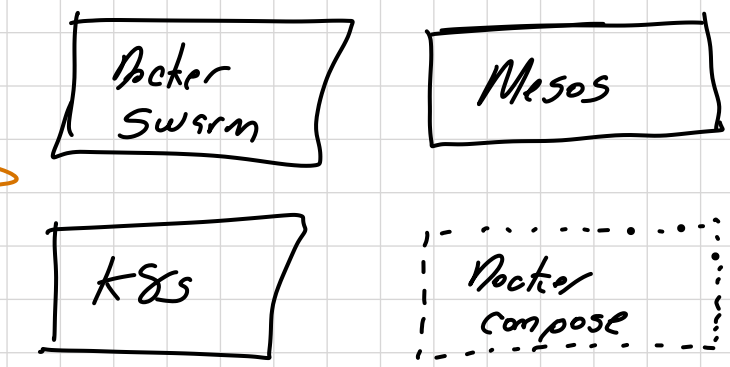
Pipeline básico:



* Uso de una imagen base
* Agregar dependencias
* Agregar aplicación

Tareas adicionales:
* Tests
* Integración
* Automatización

---- Orquestador de Contenedores ----



* El contenedor expone puertos hacia el equipo host. (iptables)

Práctica:

- Descarga de imágenes
- Cocinar una imágen
- Registries
- Ejecución de contenedores

Tareas básicas:

* Conectandose a un contenedor
* Revisión de logs
* Administración de contenedores
* Administración de imágenes

Comandos:

- docker search
- docker pull
- docker run
- docker exec
- docker logs
- docker images
- docker volumes
- docker system prune

Recomendaciones:

- imagenes livianas (capas)
- inmutables (no cambios al contenedor)
- Implementar CI/CD
- Pruebas
- Auditorias de vulnerabilidades
- Control de versiones