DOCKER SWARM

ORQUESTANDO CONTENEDORES EN LA NUBE

Creada por Alejandro Escanero Blanco / Twitter: @aescanero

Documentación y demo en
https://github.com/aescanero/dockerevents/opensouthcode

INTRODUCCIÓN A DOCKER

¿QUE ES CONTENEDOR?

ES UN PROCESO QUE EJECUTA AISLADO SU PROPIO ESPACIO DE MEMORIA, CPU, I/O Y RED

EN LINUX SE UTILIZAN DOS CARACTERISTICAS PARA ELLO: NAMESPACES AN CGROUPS

EXISTEN MUCHAS IMPLEMENTACIONES EN EL MERCADO: DOCKER, LXC, RKT, OPENVZ...

Fuente: What even is a container: namespaces and cgroups

Fuente: Cgroups, namespaces, and beyond: what are containers made from?

EXISTE UN GRAN ESFUERZO DE ESTANDARIZACIÓN

OPEN CONTAINER INITIATIVE: DEFINE LAS ESPECIFICACIONES DEL MOTOR DE EJECUCIÓN E IMAGENES DE LOS CONTENEDORES

CLOUD NATIVE COMPUTING FOUNDATION: BAJO EL PARAGUAS DE LA LINUX FOUNDATION DEFINE LAS TECNOLOGÍAS DE CONTENEDORES Y CLOUD

Fuente: Native Landscape (CNCF and OCI)

¿QUE ES DOCKER?

ES UN PROYECTO OPEN SOURCE: MOBY PROJECT

ES UN PRODUCTO ORIENTADO A LA COMUNIDAD: COMMUNITY EDITION

ES UN PRODUCTO ORIENTADO A ENTORNOS EMPRESARIALES: ENTERPRISE EDITION

Fuente: INTRODUCING MOBY PROJECT

Fuente: ANNOUNCING DOCKER ENTERPRISE EDITION

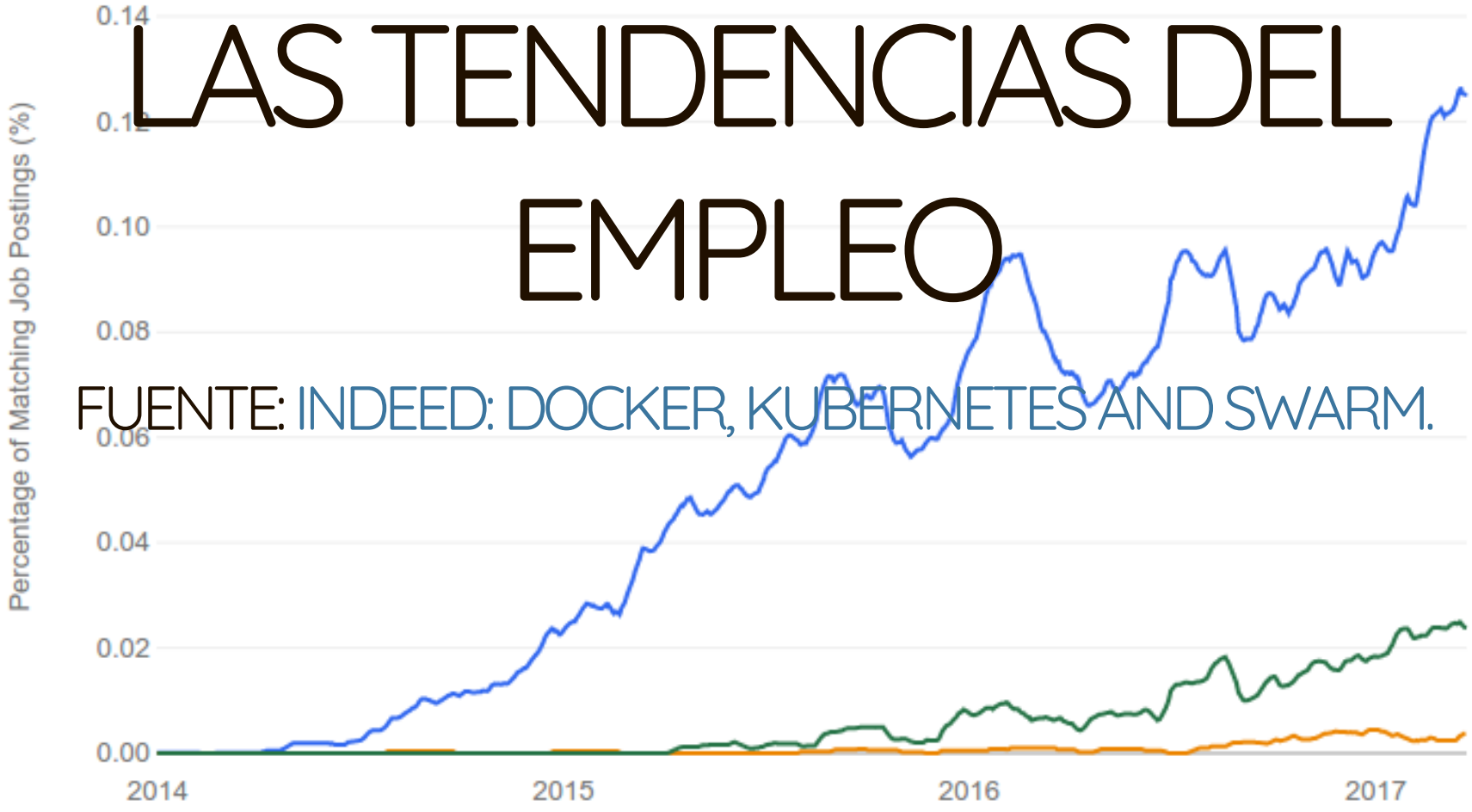
PERO SOBRE TODO ES UNA TENDENCIA DEL MERCADO

docker × swarm × kubernetes × + Add Term Find Trends

Ofertas de Empleo

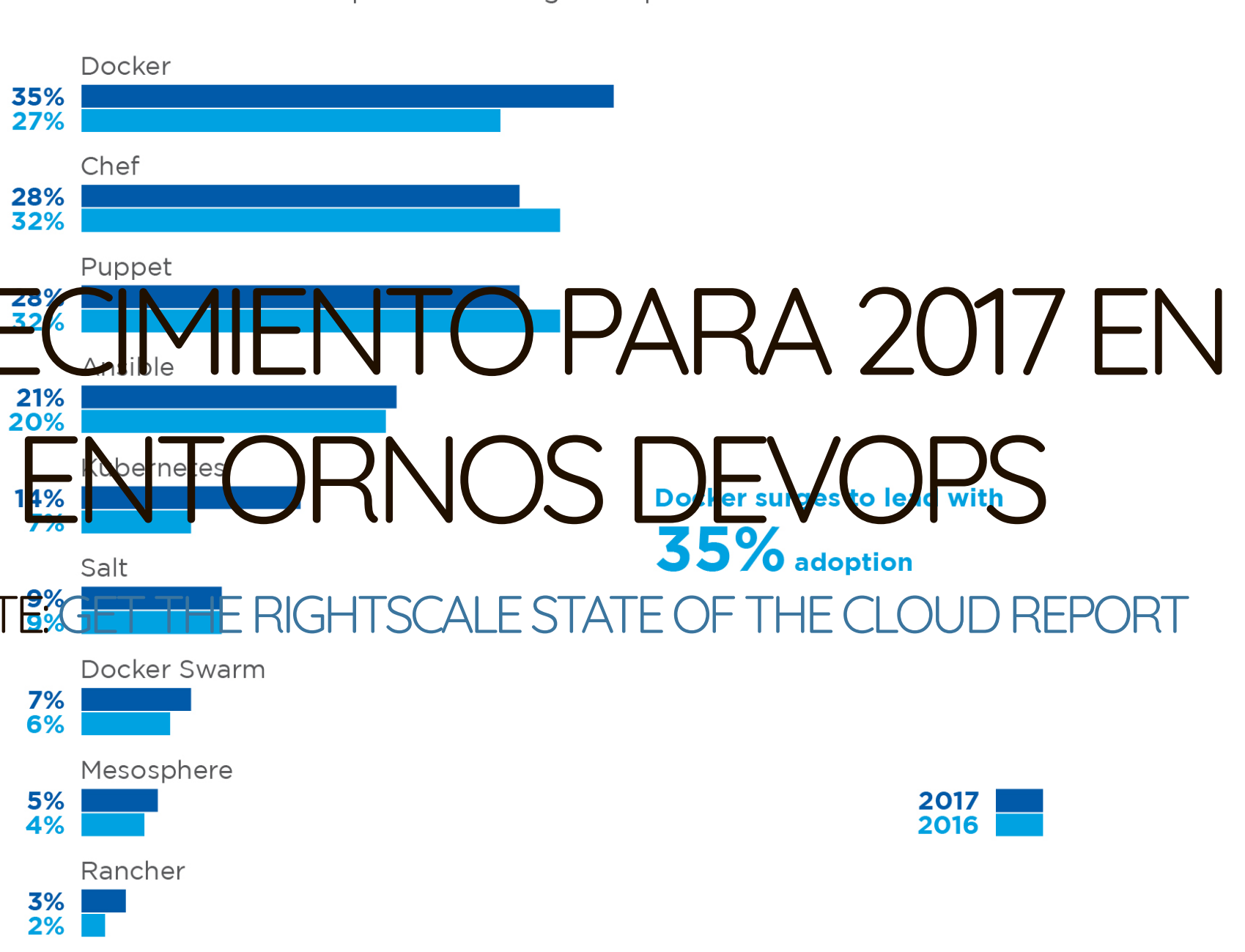
LAS TENDENCIAS DEL EMPLEO

FUENTE: INDEED: DOCKER, KUBERNETES AND SWARM.



CRECIMIENTO PARA 2017 EN ENTORNOS DEVOPS

FUENTE: GET THE RIGHTSCALE STATE OF THE CLOUD REPORT



end #5: Containers Get Orchestration

Source: RightScale 2016 State of the Cloud R

INTERES EN 2016 EN ENTORNOS DEVOPS

FUENTE: DOCKER TRENDS

FUENTE: CLOUD TRENDS FOR 2017 AND ACTIONS YOU CAN TAKE NOW

CHARLAS RELACIONADAS CON DOCKER EN OPENSOUTHCODE

DESARROLLANDO CON DOCKER

SALA FUENGIROLA 16:00

INTEGRACIÓN CONTÍNUA DE APLICACIONES MÓVILES CON DOCKER Y APPIUM

SALA FUENGIROLA 17:00

CUANDO DEV CONOCIÓ A OPS

SALA FUENGIROLA 18:00

ZERO DOWNTIME APPLICATIONS WITH OPENSHIFT

SALA RIOGORDO 13:00


Now what?

NUBES DE CONTENEDORES

DESARROLLOS MAS RÁPIDOS, CON MAYOR LIBERTAD PARA LOS DESARROLLADORES

ENTORNOS MAS COMPLEJOS, DENSOS Y CAMBIANTES

ENTORNOS DE TECNOLOGÍA DISPARES Y DISTRIBUIDOS

POLITICAS Y SEGURIDAD EMPRESARIAL

...Y PRODUCCIÓN...

Fuente: Top 5 challenges with deploying containers in production

e landscape
en source predominates)

ntainer
chestration

RANCHER

kubernetes

CoreOS

mesosphere

ntainer
ines

docker

Rocket

Fuente: DevOps, Microservices and containers - a high level overview

ni OSs

RANCHEROS

ATOMIC

PHOTON

CoreOS

Nano Server

uds with
cker support

openstack

JOYENT TRITON

OPENSHIFT

CLOUD FOUNDRY

EC2

Azu


DOCKER
SWARM

EL MODO DE GESTIÓN DE CLUSTER DISTRIBUIDO DE DOCKER: SWARM

Los Engine ("Nodos" en modo Swarm) Docker se configuran para distribuir la carga y tendremos dos tipos.

Aquellos Nodos que se mantienen la configuración de todos los servicios Docker son los "Manager", se configuran para que se comuniquen entre ellos usando el protocolo Raft

Uno de los Managers será elegido lider, mientras que el resto son elegibles.

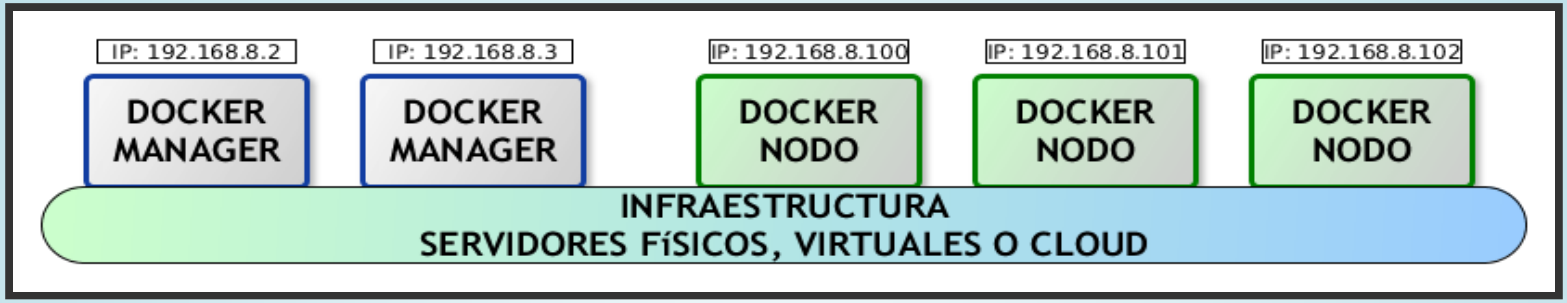
Todos los Nodos (incluso los Managers) reciben y ejecutan tareas enviadas por los Manager

Fuente: Raft consensus in swarm mode

Fuente: DevOps, Microservices and containers - a high level overview

Fuente: Swarm mode key concepts

EJEMPLO



Iniciando el primer nodo manager en la IP 192.168.8.2, nos devolverá un "token" que usaremos para conectar los nodos

```
docker swarm init --advertise-addr "192.168.8.2" --listen-addr "192.168.8.2:2377"
```

Procedemos a consultar que "token" es necesario para añadir nuevos Managers

```
docker swarm join-token manager
```

Añadimos un nuevo manager con la IP 192.168.8.3

```
docker swarm join --advertise-addr "192.168.8.3" --listen-addr "192.168.8.3:2377" --token $key 192.1
```

Añadimos el resto de nodos no Manager

```
docker swarm join --token $key 192.168.8.2
```

Un resultado de un conjunto de nodos con dos Manager, podemos ver cual es el lider y que los nodos están en Down (estan apagados)

```
swarm-master-1:~$ sudo docker node ls
ID                           HOSTNAME        STATUS  AVAILABILITY  MANAGER STATUS
1jl7pgsx23vpxnet7lvsyq9yf    swarm-node-3    Down    Active
5p00x5dn53xfz33v8ad1jwiyq    swarm-node-2    Down    Active
mjxf52vga4t47ns92ym7pq8xq    swarm-node-1    Down    Active
o7az8ubf8sk8yb9jp18gz53w8 *  swarm-master-1  Ready   Active        Reachable
q19pfkrgd2nphqmj1yd0wp2yz    swarm-master-2  Ready   Active        Leader
```

GESTIONANDO LAS IMAGENES DE LOS CONTENEDORES: REGISTRY

LOS CONTENEDORES SE GENERAN DESDE IMAGENES, Y DESPUES INSTALAR EL SERVICIO QUE NECESITEMOS

EL REGISTRO ES UNA HERRAMIENTA QUE NOS PERMITE GUARDAR LAS IMAGENES QUE YA HALLAMOS CREADO O DESCARGADO

DEBEMOS TENER UN REGISTRO ES NUESTRA NUBE PARA EVITAR QUE CADA NODO VAYA A INTERNET A DESCARGAR LAS IMAGENES

REDES DENTRO DE LA NUBE

DENTRO DEL MODELO DE RED DE CONTENEDORES, LOS CONTENEDORES MANTIENEN UNA RED QUE LOS CONECTA AL DIRECTAMENTE NODO (RED HOST)

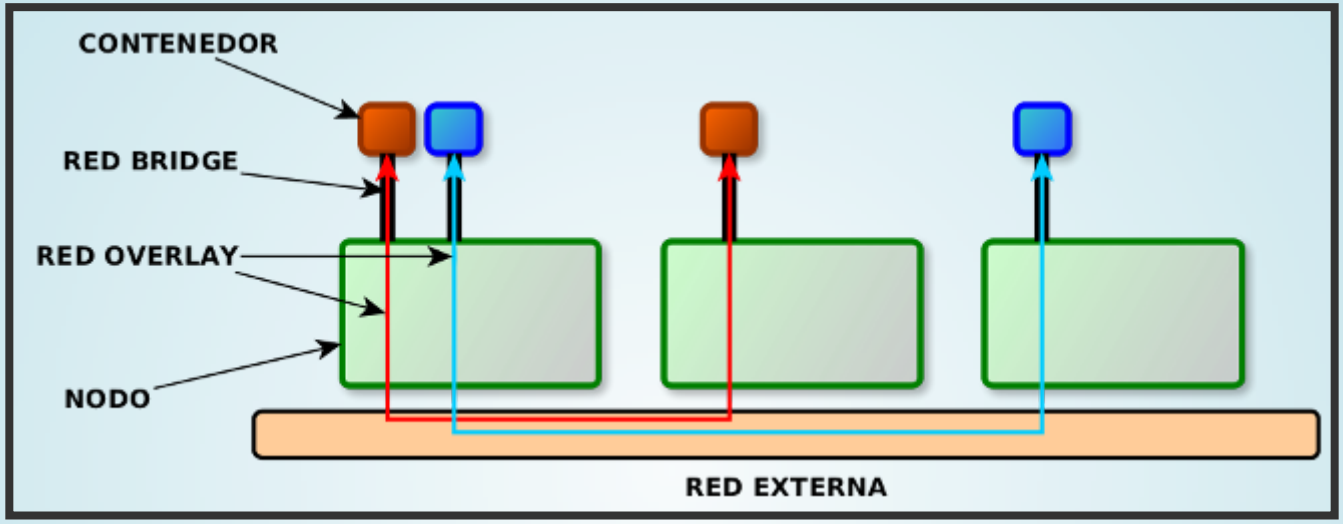
PARA ACCEDER AL EXTERIOR, SE UTILIZA UNA RED DE PUENTE (BRIDGE).

UTILIZANDO LA RED BRIDGE Y VXLAN SE PUEDEN CONSEGUIR REDES QUE CONECTEN UNOS CONTENEDORES CON OTROS

ESTAS REDES SE CONOCEN COMO REDES OVERLAY

Fuente: Docker Reference Architecture: Designing Scalable, Portable Docker Container Networks

EJEMPLO



Creamos dos redes overlay

```
sudo docker network create -d overlay be
sudo docker network create -d overlay fe
```

Comprobamos que efectivamente se despliegan

```
sudo docker network ls
                    NETWORK ID          NAME                DRIVER              SCOPE
                    p13bh7pilb71        be                  overlay             swarm
                    cca979284ca2        bridge              bridge              local
                    abedd882c417        docker_gwbridge     bridge              local
                    2k04q42d8l9v        fe                  overlay             swarm
                    0fab68cf1222        host                host                local
                    jsh8aaw7nebr        ingress             overlay             swarm
                    3fba02d48df4        none                null                local
```

SE PUEDEN AÑADIR OTROS TIPOS DE REDES (NETWORK PLUGINS)

CONTIV

ES UNA EXTENSIÓN (PLUGIN) CREADA POR CISCO, IDEADA PARA INTEGRAR LAS REDES DE SOLUCIONES DE CLOUD HETEROGENEAS CON GESTIÓN DE POLÍTICAS

WEAVE

PERMITE LA INTEGRACIÓN DE REDES DE CLOUD HETEROGENEAS, INTEGRA SU PROPIO SISTEMA DE DESCUBRIMIENTO E IPAM.

Fuente: Contiv Features

Fuente: Introducing Weave Net

LOS SERVICIOS O COMO SE DESPLIEGAN LOS CONTENEDORES EN SWARM.

EL SERVICIO ES UN EMPAQUETADO DE CONTENEDORES QUE ADEMAS INCLUYE UNA SERIE DE REGLAS

EL PUERTO DONDE ES VISIBLE EL SERVICIO, LA RED A LA QUE SE CONECTA, RESERVAS DE CPU Y MEMORIA

POLITICAS DE DESPLIEGUE, DISPONIBILIDAD Y NÚMERO DE RÉPLICAS

Fuente: How services work

COMPORTAMIENTO DE LOS SERVICIOS EN LA RED (ENDPOINT MODE)

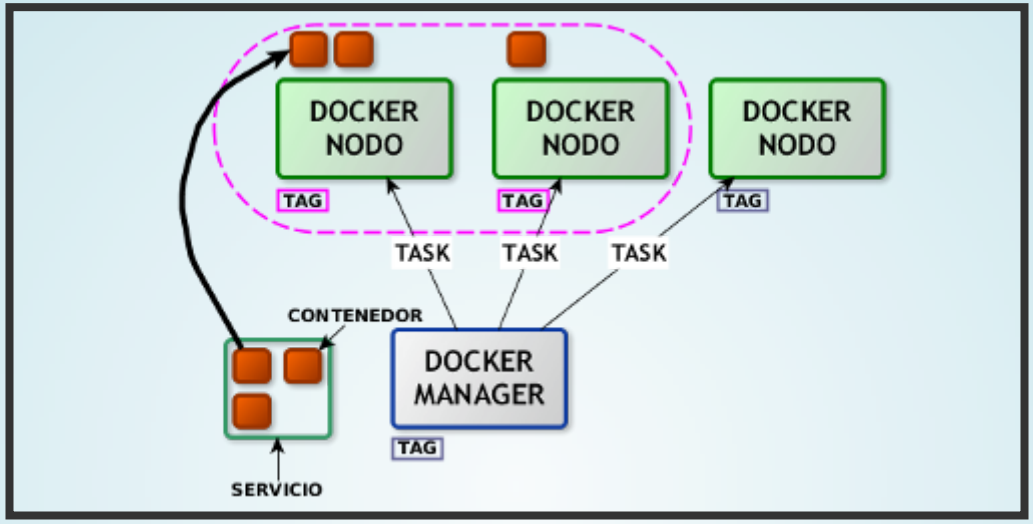
VIP (VIRTUAL IP): SE ASIGNA UNA IP PARA EL SERVICIO, Y A CADA CONTENEDOR UNA IP DEL MISMO RANGO

LA IP DEL SERVICIO ES VIRTUAL Y SIRVE PARA BALANCEAR ENTRE LOS CONTENEDORES. EL NOMBRE DEL SERVICIO APUNTA A LA IP VIRTUAL

DNSRR: SE ASIGNA A CADA CONTENEDOR UNA IP DEL MISMO RANGO, EL NOMBRE DEL SERVICIO APUNTA A CADA CONTENEDOR POR ORDEN (ROUND ROBIN)

Fuente: Attach services to an overlay network

EJEMPLO



Creamos un servicio para el registro

```
sudo docker service create --constraint=engine.labels.myproject.service==fe --endpoint-mode vip \
  --network fe --publish 5000:5000 --restart-condition any --name registry registry:2
```

Comprobamos que efectivamente se despliega

```
sudo docker service list
  ID                  NAME                MODE                REPLICAS            IMAGE
  qb5wpzv0z077        registry            replicated          1/1                 registry:2
```

Comprobamos donde se despliega

```
sudo docker service ps registry
  ID                  NAME                IMAGE               NODE                DESIRED STATE
  sxuhruy385t3        registry.1          registry:2          swarm-master-2      Running
```

LOS SERVICIOS SON ESCALABLES

UN SERVICIO PUEDE ESTAR UN NÚMERO INDETERMINADO DEL MISMO CONTENEDOR YA QUE SON ELASTICOS

PODEMOS DESCUBRIR QUE OTROS NODOS FORMAN PARTE DEL SERVICIO (SERVICE DISCOVER), YA QUE DISPONE DE UN SERVICIO DNS INTEGRADO PARA CONOCER QUIEN CONFORMA EL SERVICIO Y LA DIRECCIÓN VIP.

La dirección VIP es el nombre del servicio

```
/ # ping registry
PING registry (10.0.1.2): 56 data bytes
64 bytes from 10.0.1.2: seq=0 ttl=64 time=0.094 ms
```

Los contenedores que forman el servicio se publican como tasks.SERVICIO

```
/ # nslookup tasks.registry

Name:      tasks.registry
Address 1: 10.0.1.3 8cad19f2c5dd
```

STACKS Y COMPOSER

LAS APLICACIONES ESTÁN HECHAS POR MAS DE UN ÚNICO TIPO DE CONTENEDOR

UN STACK ES UN PAQUETE DONDE SE DEFINEN LOS SERVICIOS Y REDES QUE FORMAN PARTE DE LA APLICACIÓN

COMPOSER ES UNA APLICACIÓN CAPAZ DE CONVERTIR UN GUION DONDE SE DEFINEN MULTIPLES SERVICIOS Y REDES EN UN STACK

Fuente: Overview of Docker Compose

Fuente: Compose file version 3 reference

Ejemplo

```
version: '3.1'
services:
  ldap:
    image: localhost:5000/myproject:ldap_v1
    deploy:
      mode: replicated
      replicas: 3
      restart_policy:
        condition: on-failure
      placement:
        constraints:
          - engine.labels.myproject.service == be
      resources:
        limits:
          cpus: '0.1'
          memory: 100M
        reservations:
          cpus: '0.01'
          memory: 50M
    networks:
      - be

  fd:
    image: localhost:5000/myproject:fd_v1
    deploy:
      placement:
        constraints:
          - engine.labels.myproject.service == fe
      resources:
        limits:
          cpus: '0.1'
          memory: 300M
        reservations:
          cpus: '0.01'
          memory: 200M
    healthcheck:
      test: ["CMD", "curl", "-f", "http://localhost"]
      interval: 1m30s
```

MONITORIZACIÓN

EXISTEN MÚLTITUD DE HERRAMIENTAS PARA GESTIONAR LAS MÉTRICAS GENERADAS POR DOCKER

UNA COMBINACIÓN DE HERRAMIENTAS EN LINEA DE COMANDOS PUEDE SER UTIL (SYSTEMD-CGTOP, DOCKER STATS), PERO CON MULTITUD DE SERVIDORES NECESITAMOS HERRAMIENTAS MAS AMIGABLES.

COMO EJEMPLOS TENEMOS ELASTIC (ELK), PROMETHEUS, GRAFANA, GRAPHITE, CADVISOR, Y UN SIN FIN.

Y POR SUPUESTO LOS CLASICOS (NAGIOS, ZABBIX, SENSU, ETC), SE PUEDEN UTILIZAR PARA MONITORIZAR CONTENEDORES.

Fuente: Container Performance Analysis

Fuente: Monitoring Docker Swarm with cAdvisor, InfluxDB and Grafana

Fuente: Comparing Seven Monitoring Options for Docker

REGISTROS

AL IGUAL QUE CON LAS MÉTRICAS TENEMOS MÚLTIPLES OPCIONES PARA UNA GESTIÓN CENTRALIZADA DE LOS REGISTROS DE LOS SERVICIOS

AL CREAR EL SERVICIO SE PUEDE CONFIGURAR PARA QUE UTILICE UN SYSLOG REMOTO, AUNQUE HAY OTRAS OPCIONES COMO FILEBEAT PARA EL ENVIO DE REGISTROS

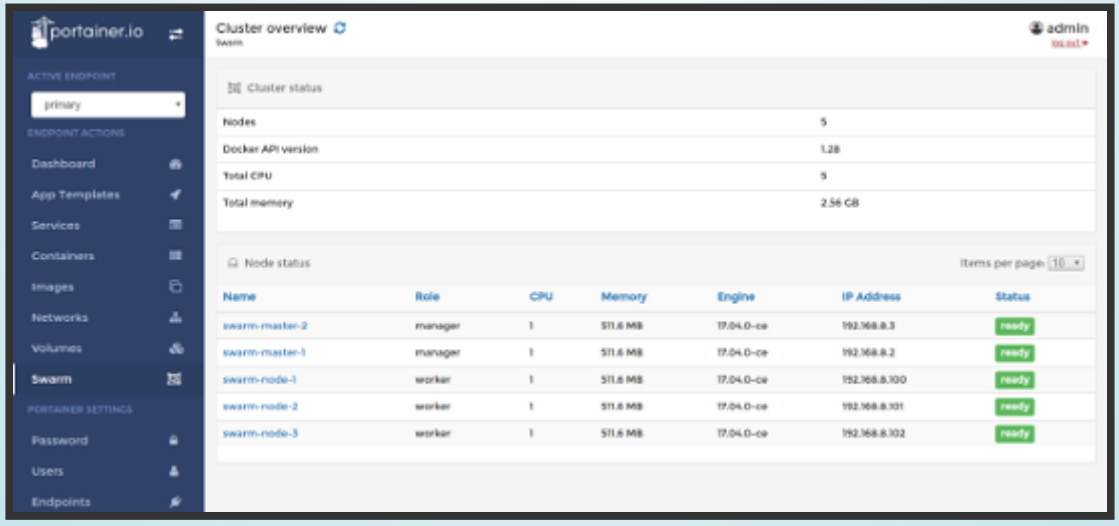
DOCKER DISPONE DE PLUGINS PARA ENVIOS DE REGISTROS A SOLUCIONES COMO GRAYLOG2, SPLUNK Y FLUENTD ENTRE MUCHAS.

Fuente: Configure logging drivers

ENTORNOS AMIGABLES

PARA ENTORNOS DE PEQUEÑO Y MEDIANO TAMAÑO PODEMOS UTILIZAR INTERFACES QUE FACILITEN NUESTRO TRABAJO, COMO POR EJEMPLO SON PORTAINER, SHIPYARD O RANCHER.

PARA GRANDES ENTORNOS DEBEMOS USAR HERRAMIENTAS DE ORQUESTACIÓN COMO SON PUPPET, ANSIBLE O CHEF.



SECCIÓN DE OPINIÓN

TODOS LOS PRODUCTOS DE DOCKER NO ESTÁN CARENTES DE DUDAS SOBRE SU ESTABILIDAD Y FUTURO.

EN MOBY/DOCKER IN PRODUCTION: A HISTORY OF FAILURE CUESTIONAN SI DOCKER ES VÁLIDO PARA PRODUCCIÓN, POR SUS PROBLEMAS DE ESTABILIDAD, DE COMPATIBILIDAD Y DE VERSIONES.

EN 9 CRITICAL DECISIONS FOR RUNNING DOCKER IN PRODUCTION HABLAN DE QUE DEBEMOS TENER EN CUENTA PARA LLEVAR UN ENTORNO A PRODUCCIÓN: GESTIÓN DE IMAGENES (Y LA SEGURIDAD DE LAS MISMAS), LA RED, EL BALANCEADOR, EL DESPLIEGUE, DESCUBRIMIENTO DE SERVICIOS, GESTIÓN DE REGISTROS, MONITORIZACIÓN Y BASES DE DATOS.

DUDAS Y CONSULTAS

RECOMENDACIONES:

CANAL DE YOUTUBE DE DOCKER

MEETUP DOCKER SEVILLA