En esta entrada vamos a introducir la funcionalidad Swarm que soporta el motor (engine) de Docker, esto nos permitirá desplegar contenedores en varias máquinas añadiendo una infraestructura clúster entre ellas, y así poder utilizar y compartir los recursos de varias máquinas con Docker como si fuera una única instancia.

La gestión de red y algunos recursos es transparente para el usuario, el Swarm se encarga de *orquestar* la comunicación entre contenedores, también se publican los puertos compartidos de los contenedores por defecto en todos los nodos, por lo que no tendremos de que preocuparnos, da la posibilidad de escalar servicios fácilmente y distribuirlos entre los nodos, etc.

Iniciar el Swarm

En el Swarm de Docker existen dos tipos de nodos, los manager y los worker, siendo el manager el que delega las tareas a los workers y no al revés, el manager también forma parte del Swarm. En un entorno en producción se recomienda tener mínimo unos *tres* managers para soportar la tolerancia a fallos.

Se inicia el swarm en el manager con el siguiente comando

```
$ docker swarm init
```

Para añadir un manager al Swarm se ejecuta el siguiente comando, hay que seguir las instrucciones en el nodo correspondiente

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

Para añadir un worker se ejecuta y seguir las instrucciones

```
$ docker swarm join-token worker
```

Una vez añadidos los nodos podremos visualizarlos desde el manager ejecutando

```
$ docker node ls
```

Desplegar servicios

Para desplegar (deploy) servicios en el Swarm usaremos docker-compose y la configuración correspondiente que está disponible a partir de la *versión 3*, configuramos un servicio:

```
version: '3'

services:
  servicio:
    image: imagen_del_servicio
    deploy:
      mode: replicated
      replicas: 1
      restart_policy:
        condition: on-failure
      placement:
        constraints: [node.hostname==nodehostname]
    volumes:
      - ssh-entrypoint:/entrypoint:ro
```

```
    networks:
      netservice:

networks:
  netservice:

volumes:
  ssh-entrypoint:
    driver: vieux/sshfs
    driver_opts:
      sshcmd: user@host:/remote_shared_folder
      password: secret
      port: 2222
      sftp_server: "/usr/bin/sudo /usr/lib/openssh/sftp-server"
      allow_other: ""
```

Ejecutamos el stack que vamos a desplegar en el Swarm

```
$ docker stack up --with-registry-auth -c docker-compose.yml
nombre_del_servicio
```

La opción *–with-registry-auth* se utiliza por si usamos imágenes de un registro de Docker propio, el Swarm buscará la imagen a desplegar en los registros que estemos conectados en los nodos y la bajará en cada uno de ellos. Pueden mezclarse arquitecturas para tener un conjunto de imágenes multiarch jugando con el manifest del repositorio, pero esa opción es todavía experimental.

Hemos creado el volumen compartido por sshfs con el driver vieux/sshfs para varias arquitecturas, por lo que cada nodo se conectará a la carpeta /remote_shared_folder del host que indiquemos con la correspondiente configuración por sshfs.

En el apartado node.hostname se indica una máquina concreta para desplegar el servicio, si se omite esta opción se replicará en la que elija el Swarm

Gestión del Swarm

Una vez desplegado el servicio puede verse ejecutando

```
$ docker service ls
```

Y si queremos ver uno en concreto

```
$ docker service ps nombre_del_servicio
```

Para ver logs de un servicio se ejecuta

```
$ docker service logs -f nombre_del_servicio
```

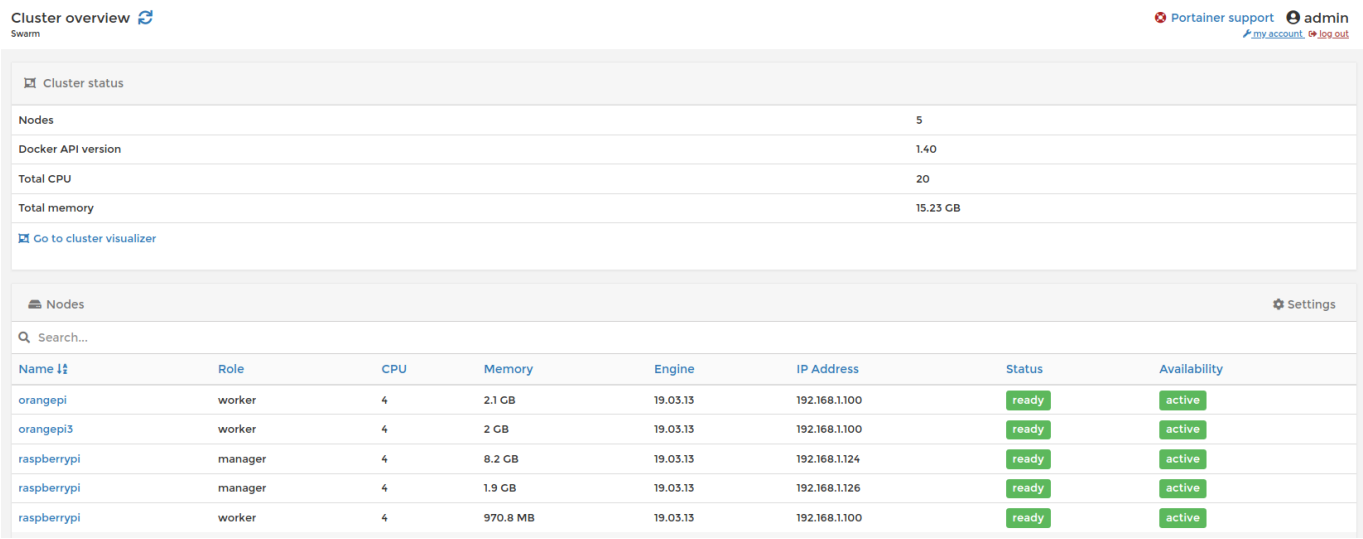
Eliminar un servicio ejecutando

```
$ docker stack rm nombre_del_servicio
```

Y eliminar un nodo del Swarm

```
$ docker node rm nodo_a_eliminar
```

Por último comentar que podemos usar otras herramientas para gestionar el Swarm como es Portainer, aunque existen otras muchas nos facilita información gráficamente del clúster.



Esto es todo por el momento, volveremos pronto con Kubernetes (k8s), saludos