sesión 13

Container Orchestration Rosetta Stone

	unidad mínima	Gestor de disponibilidad controllers	expose services	webservices exterior
Swarm	contenedor	Docker Service		
K8S	pod	~~Replication Controller~~ ReplicaSet DaemonSet StatefulSet	HostPort LoadBalancer	LoadBalancer Ingress

- controllers
  - Deployments = POD + ReplicaSet
  - Jobs = 1 sola ejecución
  - CronJob
- services
  - hostport : exponer en el host a través de un puerto
  - clusterIP : uso interno del cluster
  - loadbalancer :
  - externalname : definimos el mismo nombre en el servicio en diferentes namespaces, ex: oracle (dev/pre/pro)
- webservices (acceso exterior):
  - Loadbalancer
  - ingress : trabaja en kube-system, pero expone de cualquier namespace (podría usarse para hacer hablar PODs de diferentes namespaces)
  - nginx+ssl, traefik

lab

levantar docker swarm cluster, desplegar Portainer y ELK

portainer.yaml

```
version: '3.4'

services:
  portainer:
    image: portainer/portainer
    command: '-H "tcp://tasks.agent:9001" --tlsskipverify'
    ports:
      - ${PUBLIC_PORT}:9000
    networks:
      - proxy
      - portainer_agent
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
      - ${REMOTE_MOUNT}/${STACK_NAME}/portainer/data:/data
    deploy:
      mode: replicated
      replicas: 1
      labels:
```

```
        traefik.port: 9000
        traefik.frontend.rule: "Host:${FQDN}"
        traefik.docker.network: 'proxy'
      placement:
        constraints: [node.role == manager]
  agent:
    image: portainer/agent
    environment:
      AGENT_CLUSTER_ADDR: tasks.agent
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    ports:
      - target: 9001
        published: 9001
        protocol: tcp
        mode: host
    networks:
      - portainer_agent
    deploy:
      mode: global
      placement:
        constraints: [node.platform.os == linux]



networks:
  portainer_agent:
    external: true
  proxy:
    external: true
```

elk.yaml

```
#       Docker Stack to deploy ELK + Logspout
#       Based on .......
#       Updated by: Kenneth Peiruza, kenneth@floss.cat
#       Sun Mar  4 13:15:47 CET 2018
#
#      cluster.name: 'docker-cluster'
#      bootstrap.memory_lock: 'true'
version: '3.4'

services:
  elasticsearch:
    image: docker.elastic.co/elasticsearch/elasticsearch-oss:6.2.2
    environment:
      ES_JAVA_OPTS: '-Xms768m -Xmx768m'
      LOGSPOUT: 'ignore'
    networks:
      - elasticsearch
    volumes:
      -
/srv/docker/stack/Cluster3/elasticsearch/data:/usr/share/elasticsearch/data
    deploy:
```

```
      replicas: 1

  logstash:
    image: docker.elastic.co/logstash/logstash-oss:6.2.2
    volumes:
      -
/srv/docker/stack/Cluster3/logstash/config:/usr/share/logstash/pipeline
    depends_on:
      - elasticsearch
    networks:
      - elasticsearch
      - logstash
    environment:
      LOGSPOUT: 'ignore'
    deploy:
      replicas: 1

  logspout:
    image: bekt/logspout-logstash
    environment:
      ROUTE_URIS: 'logstash://logstash:5000'
      DOCKER_LABELS: 'yes'
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
    depends_on:
      - logstash
    networks:
      - logstash
#    environment:
#      LOGSPOUT: 'ignore'
# OJO, se ignora a si mismo y no se vuelcan logs
    deploy:
      mode: global
      restart_policy:
        condition: on-failure
        delay: 30s

  kibana:
    image: docker.elastic.co/kibana/kibana-oss:6.2.2
    ports:
      - 5601:5601
    depends_on:
      - elasticsearch
    networks:
      - elasticsearch
      - proxy
    environment:
      ELASTICSEARCH_URL: 'http://elasticsearch:9200'
      LOGSPOUT: 'ignore'
    deploy:
      replicas: 1
      labels:
        traefik.port: 5601
        traefik.frontend.rule: "Host:logs.local"
        traefik.docker.network: "proxy"
```

```
networks:
  default:
    driver: 'overlay'
  logstash:
    driver: 'overlay'
  elasticsearch:
    driver: 'overlay'
  proxy:
    external: true
```

var

```
REMOTE_BIND=/srv/docker/stack
STACK_NAME=Cluster3
LOGS_URL=logs.local
```

logstatsh.conf

```
input {
  udp {
    port  => 5000
    codec => json
  }
}

filter {
  if [docker][image] =~ /logstash/ {
    drop { }
  }
}

output {
  elasticsearch { hosts => ["elasticsearch:9200"] }
}
```

https://logz.io/blog/logstash-grok/

procesar en ELK

logspout → logstash → kibana

1. debemos habilitar las DOCKER_LABELS como variable de entorno en el deploy de la imagen (cualquier valor lo activa)

   ejemplo.yaml

   ```
   version: '3'

   services:
      db:
        image: mysql:5.7
        volumes:
   ```

```
      - db_data:/var/lib/mysql
    restart: always
    environment:
      MYSQL_ROOT_PASSWORD: ${MYSQL_DATABASE_PASSWORD}
      MYSQL_DATABASE: wordpress
      MYSQL_USER: wordpress
      MYSQL_PASSWORD: wordpress

  wordpress:
    image: wordpress:latest
    ports:
      - 80
    restart: always
    environment:
      WORDPRESS_DB_HOST: db:3306
      WORDPRESS_DB_USER: wordpress
      WORDPRESS_DB_PASSWORD: wordpress
    deploy:
      labels:
        logformat: 'combined'

volumes:
    db_data:
```

2. añadimos una etiqueta en nuestro contenedor para poder discriminar su origen (y formato)
3. modificamos el logstash.conf para añadir un **if** por etiqueta y decirle que formato tiene

```
if [docker][image] =~ /^cadena/ {
  grok {
    match => {"message" => "COMBINEDAPACHELOG"}
  }
}
```

   - https://nathanleclaire.com/blog/2015/04/27/automating-docker-logging-elasticsearch-logstash-kibana-and-logspout/
   - https://www.elastic.co/guide/en/logstash/current/event-dependent-configuration.html
   - https://discuss.elastic.co/t/help-with-multiple-if-else-if-configuration/40417

otros

- whitedisplay.com
- grok debugger
- vim
  - borrar lineas de comentarios y vacías:
    - junto: `% g/^\(#|\$)/d`
    - comentarios: `% g/^#/d`
    - líneas vacías: `% g/^$/d`