

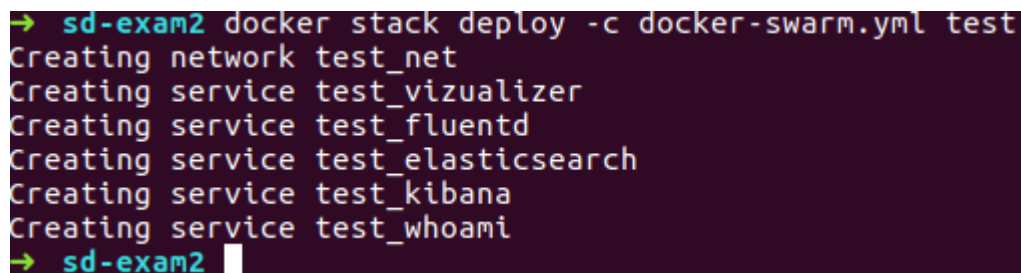
Examen 2: Sistemas distribuidos
Christian David Cárdenas Camargo
A00212740

Prerrequisitos: docker, docker-engine, docker-swarm

Resumen: Se realizará el despliegue de el servicio whoami y visualizer haciendo uso de docker swarm para para que el despliegue se haga de forma remota. Además, se hará uso de elasticsearch, fluentd y kibana para el registro de logs y su visualización. Los logs se presentarán dentro de Kibana de forma que la información sea explícita mediante un filtro aplicado en el archivo de configuración de fluentd.

1. Como se va a utilizar docker swarm, es necesario definir un nodo manager y otros nodos en modo worker, el nodo manager se el administrador del cluster mientras los trabajadores hacen lo que éste les indique.

```
docker swarm init --advertise-addr 192.168.130.128
docker service ls
docker build -t cdcardenas/myfluentd:latest .
docker push cdcardenas/myfluentd
docker stack deploy -c docker-swarm.yml test
```



```
→ sd-exam2 docker stack deploy -c docker-swarm.yml test
Creating network test_net
Creating service test_vizualizer
Creating service test_fluentd
Creating service test_elasticsearch
Creating service test_kibana
Creating service test_whoami
→ sd-exam2
```

el servicio web que se utilizara será whoami, para lo cual se utilizaron 2 host distintos al host manager, en dichas máquinas se ejecutó el siguiente comando para que se unieran al cluster.

```
sudo docker swarm join --token SWMTKN-1-
4ozt39js91hkv73li79tmknedmr2zjtedat4flqon9yl4zp9ng-c7brsjeh9ewc3zt8b2rgoia8
192.168.130.128:2377
```

Luego, para verificar que los servicios estén correctamente desplegados utilizamos le comando

```
docker service ls
```

revisamos que el número de réplicas sea la correcta

```

→ sd-exam2 docker service ls

```

ID	NAME	MODE	REPLICAS
5p539nj8lqcl	test_kibana	replicated	1/1
kibana:latest		*:5601->5601/tcp	
e6k75ldx509i	registry	replicated	1/1
registry:2		*:5000->5000/tcp	
f0l0uhvzwpn	test_whoami	replicated	2/2
tutum/hello-world:latest		*:80->80/tcp	
iawpty0zpxx3	test_elasticsearch	replicated	1/1
elasticsearch:latest		*:9200->9200/tcp	
n9p7akgsdrzi	test_vizualizer	replicated	1/1
dockersamples/visualizer:latest		*:8080->8080/tcp	
q533s4mtc8ik	test_fluentd	replicated	1/1
cdcardenas/myfluentd:latest		*:24224->24224/tcp, *:24224->24224/udp	

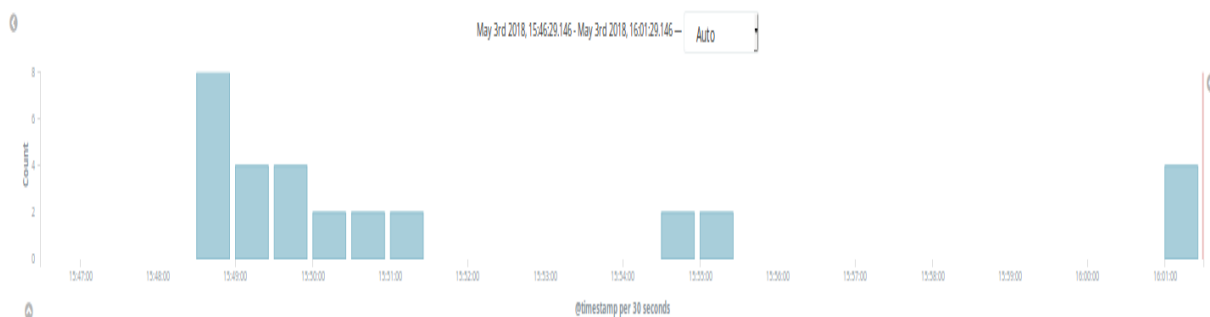
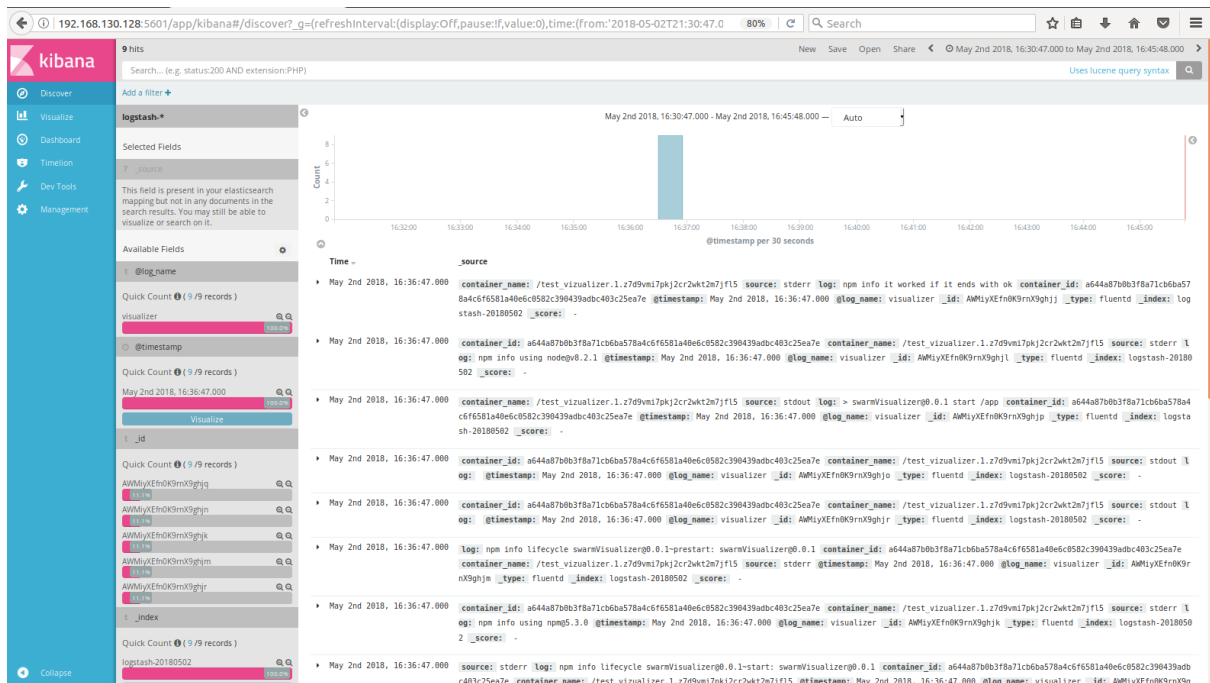
con visualizer podemos ver que servicios esta funcionando y en qué hosts.

The screenshot shows the Docker Swarm Visualizer interface. At the top, there's a search bar labeled "filter containers". Below it, the cluster is represented by three nodes: two "worker" nodes and one "manager" node, all with 7.739G RAM and x86_64/linux architecture.

The services running on the nodes are as follows:

- Left Worker Node:**
 - test_whoami:** Image: tutum/hello-world:latest, Tag: latest, Updated: 2/5 16:36, State: running.
- Manager Node:**
 - test_vizualizer:** Image: dockersamples/visualizer:latest, Tag: latest, Updated: 2/5 16:37, State: running.
 - test_fluentd:** Image: cdcardenas/myfluentd:latest, Tag: latest, Updated: 2/5 16:36, State: running.
 - registry:** Image: registry:2, Tag: 2, Updated: 2/5 16:7, State: running.
 - test_elasticsearch:** Image: elasticsearch:latest, Tag: latest, Updated: 2/5 16:36, State: running.
 - test_kibana:** Image: kibana:latest, Tag: latest, Updated: 2/5 16:36, State: running.
- Right Worker Node:**
 - test_whoami:** Image: tutum/hello-world:latest, Tag: latest, Updated: 2/5 16:36, State: running.

Para el manejo de estadísticas utilizamos elasticsearch y kibana.



Para los logs de whoami se filtró en formato apache de forma que la información queda presentada de la siguiente forma

	Table	JSON
t	@log_name	tutum
@	@timestamp	May 3rd 2018, 15:51:00.000
t	_id	AwMnxcI6E5tIA0VjxJC5
t	_index	logstash-20180503
#	_score	-
t	_type	fluentd
?	agent	Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:54.0) Gecko/20100101 Firefox/54.0
?	code	304
?	host	10.255.0.2
?	method	GET
?	path	/logo.png
?	referer	http://192.168.130.128/hello
?	size	0
?	user	-

y finalmente con whoami podemos saber quien contesta la solicitud



Hello world!

My hostname is fb38101952bf

Para whoami se restringe los recursos sobre la máquina en la que se ejecuta, en este caso se le asignaron 10% de CPU y 20Mb de memoria RAM.

version: "3"

services:

```
whoami:
  image: tutum/hello-world
  networks:
    - net
  ports:
    - "80:80" # puerto de funcionamiento
  logging:
    driver: "fluentd" # Logging Driver
    options:
      tag: tutum # TAG
  deploy:
    restart_policy:
      condition: on-failure
      delay: 20s
      max_attempts: 3
      window: 120s
    mode: replicated
    replicas: 2
  resources:
    limits:
      cpus: '0.10'
      memory: 20M
    reservations:
      cpus: '0.05'
      memory: 10M
  placement:
    constraints: [node.role == worker] # Se ejecutara en los dos nodos workers
  update_config:
    delay: 2s
```

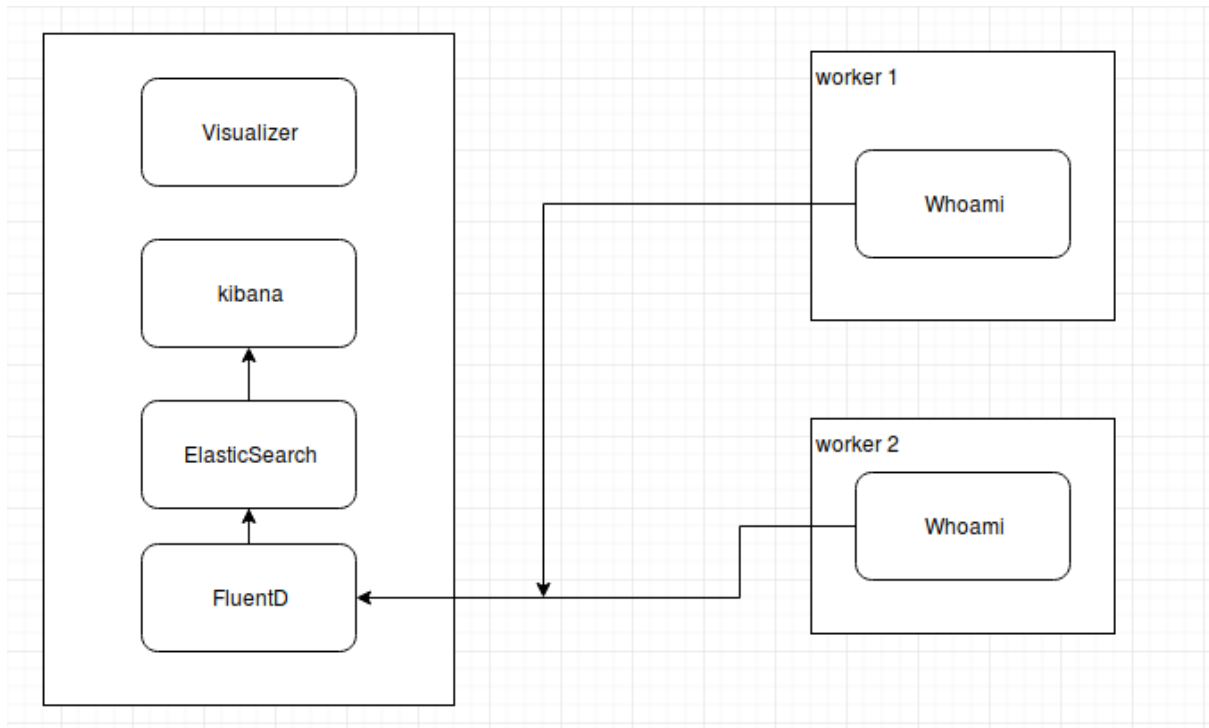
Dockerfile

```
|
# fluentd/Dockerfile
FROM fluent/fluentd:v0.12-debian
RUN echo "fluentd"
RUN ["gem", "install", "fluent-plugin-elasticsearch", "--no-rdoc", "--no-ri", "--version", "1.9.2"]
#RUN rm /fluentd/etc/fluent.conf
#COPY ./conf/fluent.conf /fluentd/etc
```

fluent.conf

```
1 <source>
2   @type forward
3   port 24224
4   bind 0.0.0.0
5 </source>
6
7 <filter *.*>
8   @type parser
9   format apache
10  key_name log
11 </filter>
12
13 <match tutum>
14   @type copy
15   <store>
16     @type file
17     path /fluentd/log/tutum.*.log
18     time_slice_format %Y%m%d
19     time_slice_wait 10m
20     time_format %Y%m%dT%H%M%S%z
21     compress gzip
22     utc
23     format json
24   </store>
25   <store>
26     @type elasticsearch
27     host elasticsearch
28     port 9200
29     logstash_format true
30     logstash_prefix logstash
31     logstash_dateformat %Y%m%d
32     include_tag_key true
33     tag_key @log_name
34     flush_interval 1s
35   </store>
36   <store>
37     @type stdout
38   </store>
39 </match>
40 <match visualizer>
41   @type copy
42   <store>
43     @type file
44     path /fluentd/log/visualizer.*.log
45     time_slice_format %Y%m%d
46     time_slice_wait 10m
47     time_format %Y%m%dT%H%M%S%z
48     compress gzip
49     utc
50     format json
51   </store>
52   <store>
53     @type elasticsearch
54     host elasticsearch
55     port 9200
56     logstash_format true
57     logstash_prefix logstash
58     logstash_dateformat %Y%m%d
59     include_tag_key true
60     type_name access_log
61     tag_key @log_name
62     flush_interval 1s
63   </store>
64   <store>
65     @type stdout
66   </store>
67 </match>
68
```

Diagrama UML



Problemas identificados

Se presentaron 2 inconvenientes a la hora de hacer el despliegue de los servicios, el primer problema fue que Kibana no estaba identificando el indexador y por lo tanto no se podía inicializar el servicio. Esto se soluciono cambiando el nombre del indexador a 'logstash-*'. El segundo fue fluentd no estaba le estaba enviando los Logs a elasticsearch, esto ocurrio porque el archivo de configuración de fluentd no estaba sobre escribiendo correctamente y se soluciono cargando directamente el archivo de configuración cuando se ejecutaba el comando para desplegar los servicios.