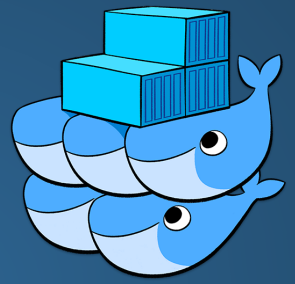
IMPLEMENTACIÓN DE

DOCKER SWARM

EN



AMAZON WEB SERVICES

USANDO

AUTO SCALING GROUPS

Y

ELASTIC LOAD BALANCING

Adrián Gómez Lois
Administración de Sistemas Informáticos y Redes
CIFP A Carballeira - Marcos Valcárcel
Noviembre - 2018

Creando un Dockerfile

- Partirá de una imagen limpia de Debian.
- Se instalarán los paquetes: net-tools, curl, mysql, apache2, php y otros paquetes necesarios relacionados: wget, nano, curl, openssh-server y supervisor.
- Creará una BBDD Worpress con usuario y password y le otorgará permisos.
- Copiará un contenido de una instalación previamente configurada con un theme personalizado y adaptado de Worpress, asignará el propietario www-data al directorio.
- Creará un directorio en el contenedor y copiará en el los certificados SSL para el dominio de la web. (Del dominio "itgal.es" adquirido a través del registrar-registrador 1&1).
- Copiará dos ficheros Virtualhost previamente configurados al servidor Apache2 del contenedor y habilitará los módulos utilizados y necesarios para Apache2.
- Creará y configurará un usuario para para la conexión SSH al contenedor.
- Establecerá un banner motd personalizado.
- Expondrá los puertos públicos para SSH, HTTP y HTTPS.
- Establecerá como entrypoint (CMD) del contenedor la ejecución de supervisord.

Subir imagen a Docker Hub

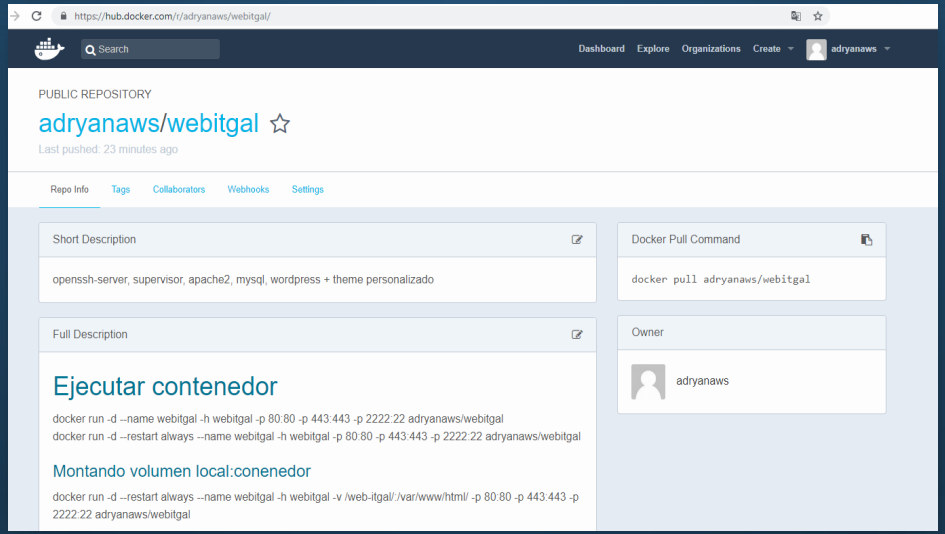
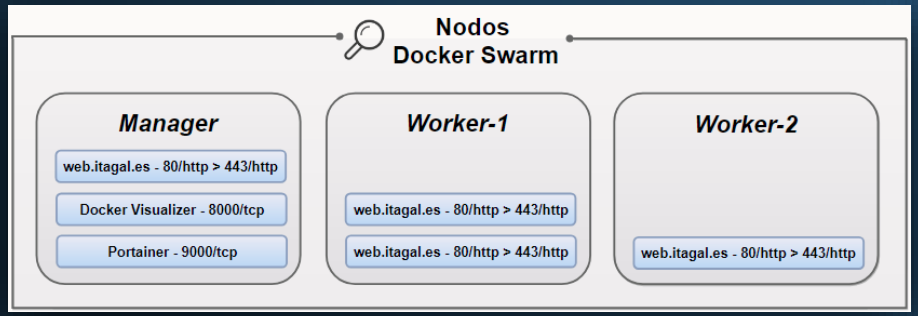
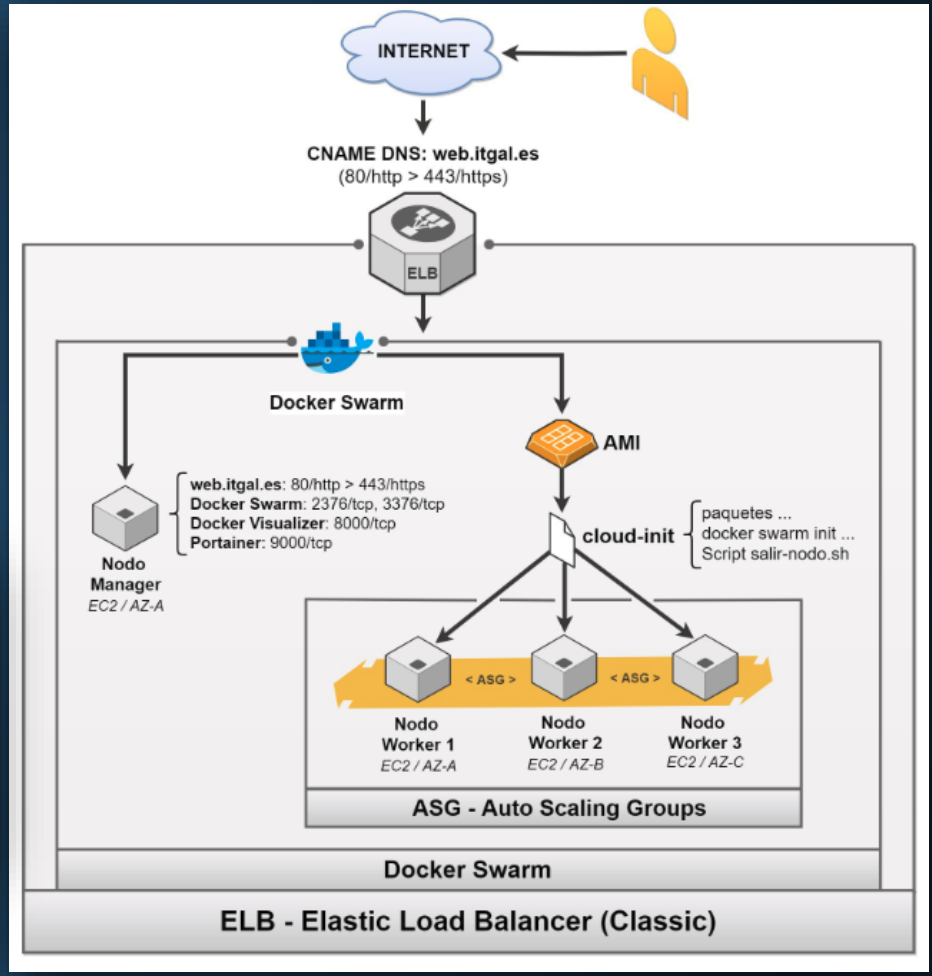


Diagrama AWS:
Docker Swarm, Auto Scaling Group, Elastic Load Balancer

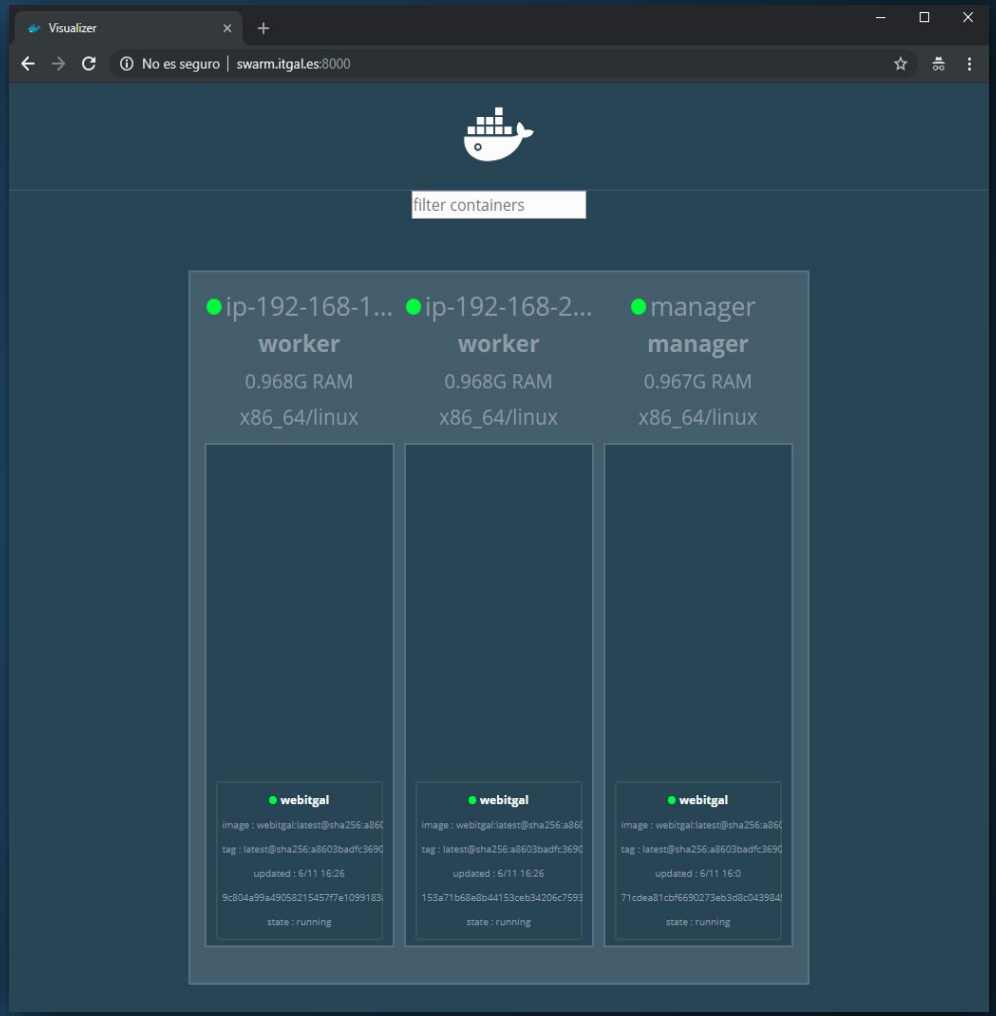




Docker Swarm:

Crear el servicio con tres tareas de réplica en tres nodos desplegados con ASG

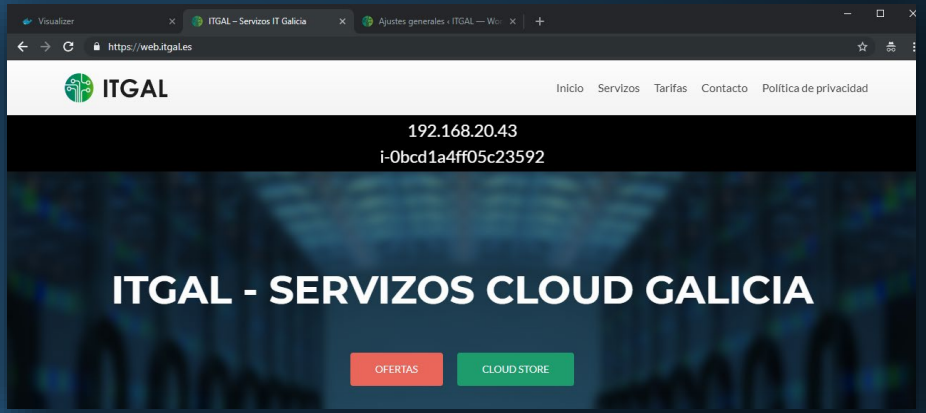
```
root@manager:~# docker service create \
> --name webitgal \
> --publish 80:80 \
> --publish 443:443 \
> --replicas 3 \
> --update-parallelism 1 \
> --update-delay 5s \
> --restart-condition on-failure \
> adryanaws/webitgal
8eoqdtcmx20mne3iepo2074px
overall progress: 3 out of 3 tasks
1/3: running   [==========================>]
2/3: running   [==========================>]
3/3: running   [==========================>]
verify: Service converged
```

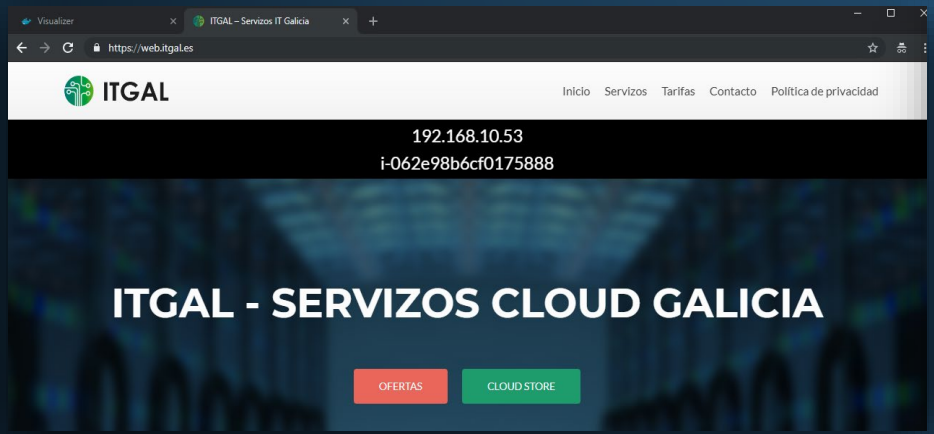


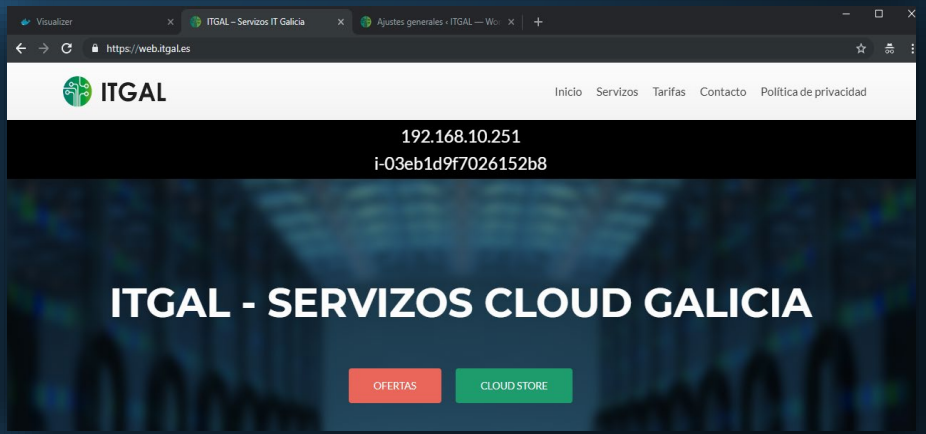
Elastic Load Balancer:

Balanceo de carga entre instancias EC2 (nodos Swarm). Para el servicio replicado “webital”

```
<div style="background-color:black;color:white;font-size:1.5em;text-align:center;padding:5px;width:100%">
<?php echo file_get_contents("http://169.254.169.254/latest/meta-data/local-ipv4"); ?>
<br />
<?php echo file_get_contents("http://169.254.169.254/latest/meta-data/instance-id"); ?>
</div>
```

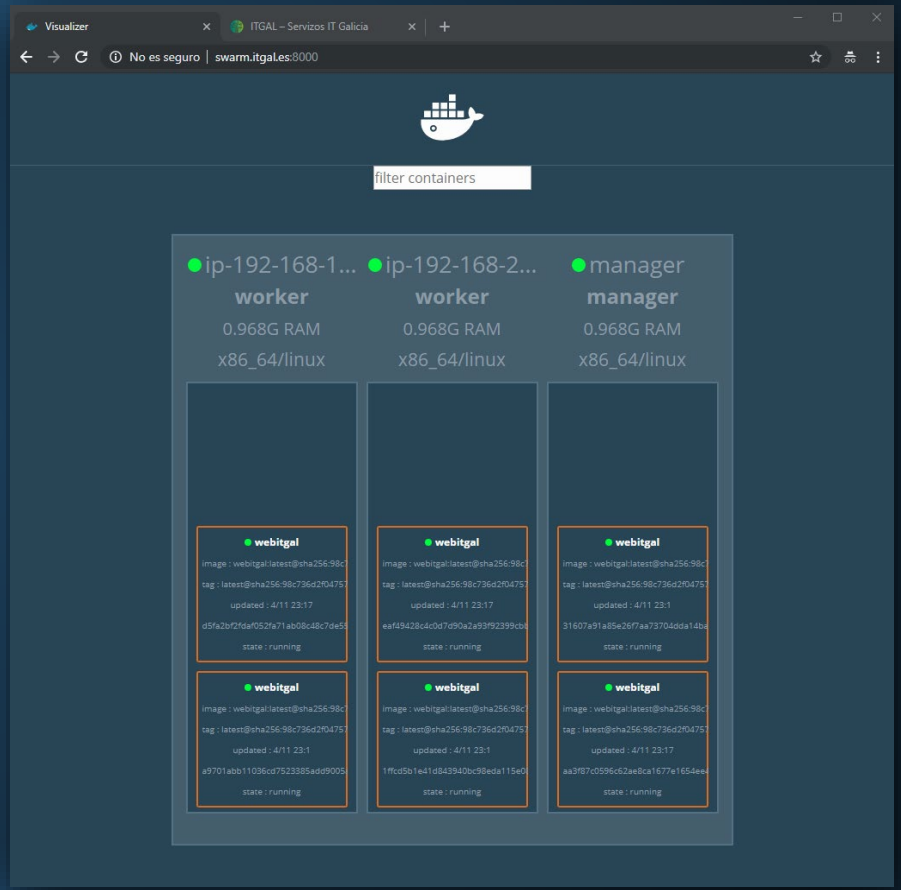






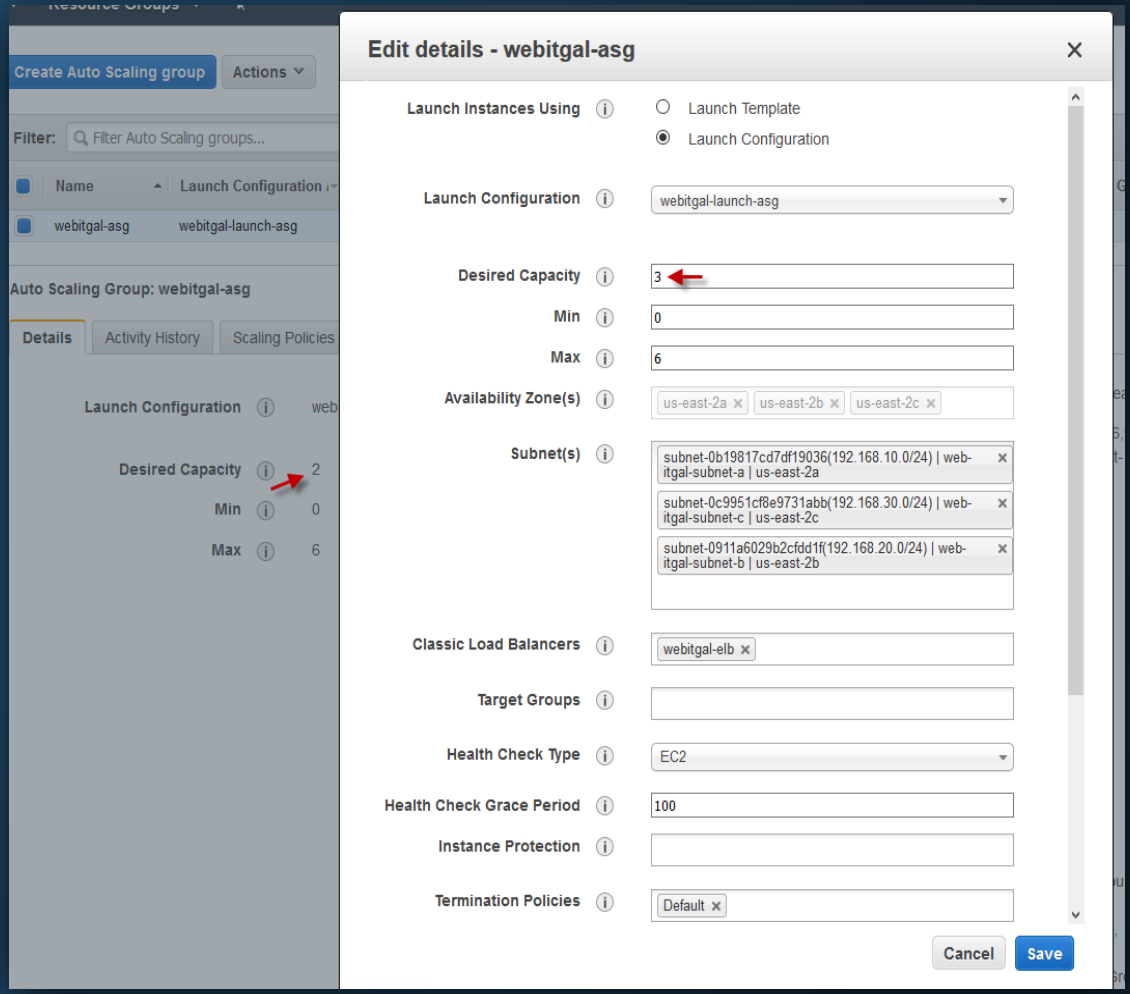
Docker Swarm:
Actualizar el servicio con tres tareas de réplica

```
root@manager:~# docker service update --replicas=6 webitgal
webitgal
overall progress: 6 out of 6 tasks
1/6: running   [==================================>]
2/6: running   [==================================>]
3/6: running   [==================================>]
4/6: running   [==================================>]
5/6: running   [==================================>]
6/6: running   [==================================>]
verify: Service converged
```



Auto Scaling Group

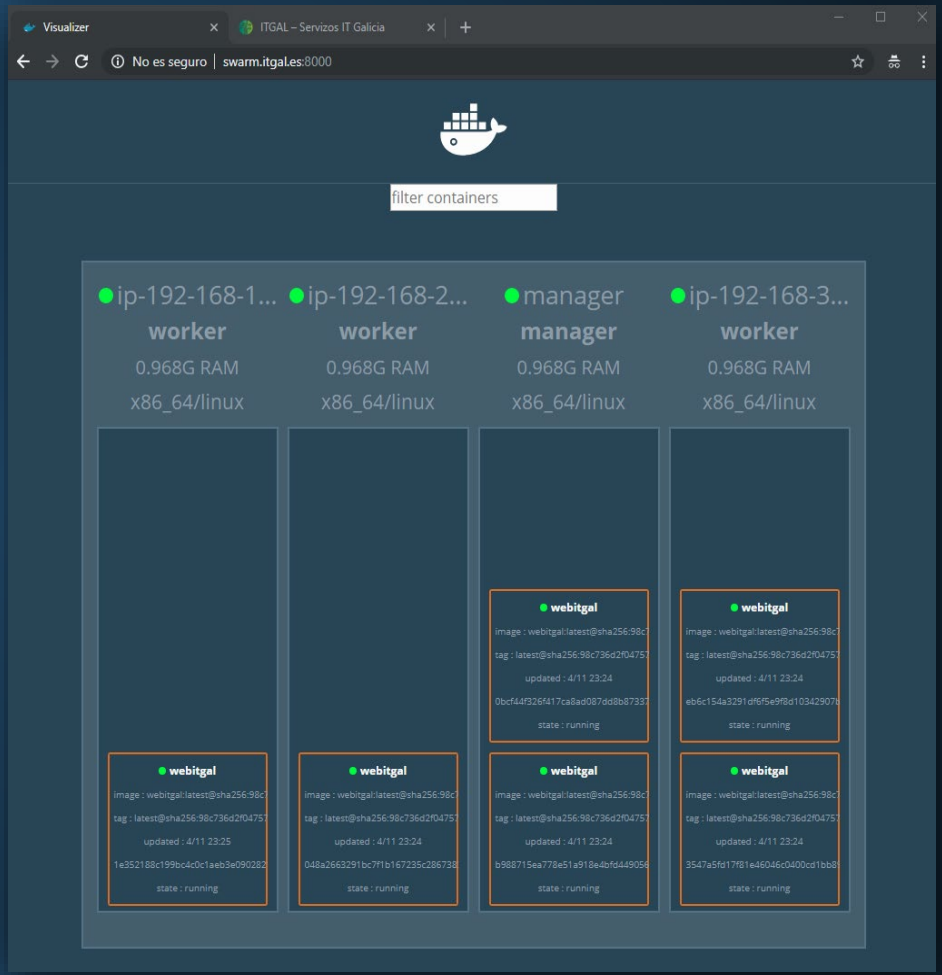
Scale-out en una instancia que se agregará al nodo Swarm



Docker Swarm:

Actualización de servicio con las misma tareas de réplicas anteriores, creando así el rebalanceo de carga entre los nodos del Swarm

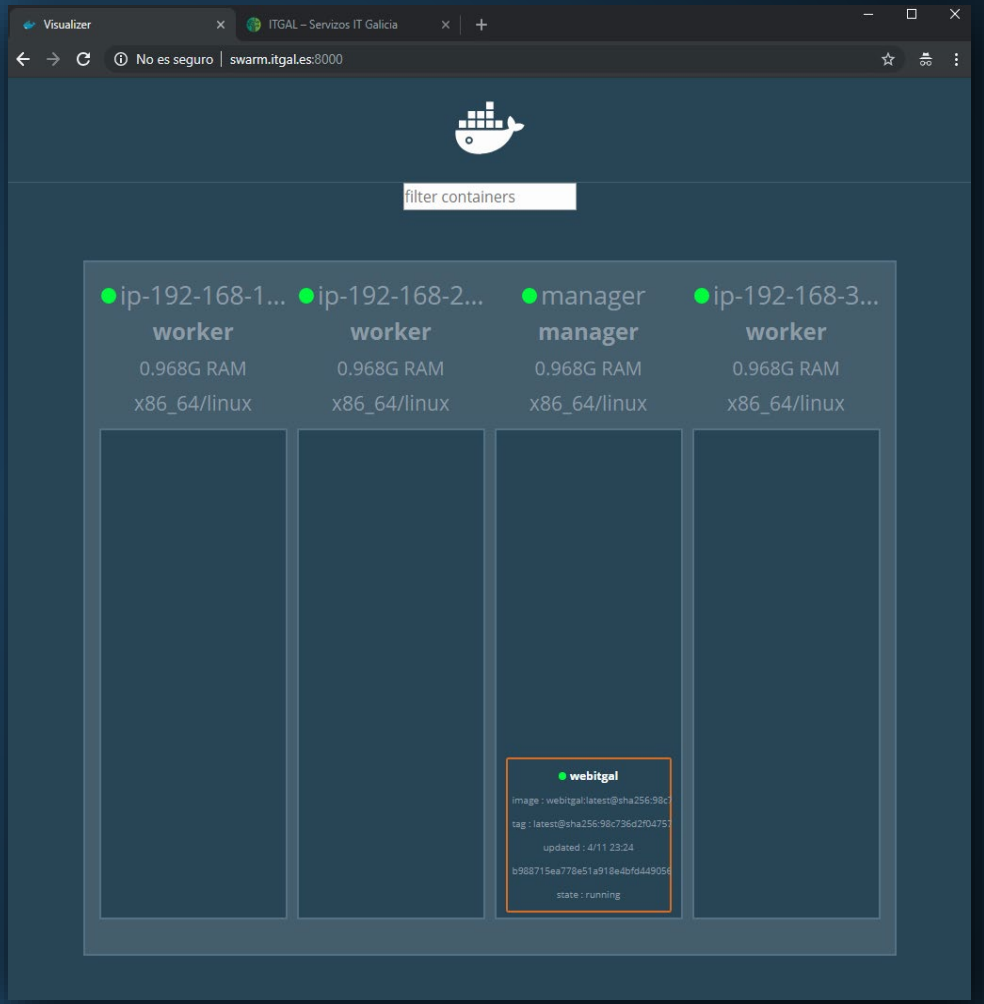
```
root@manager:~# docker service update --force webitgal
webitgal
overall progress: 6 out of 6 tasks
1/6: running   [==================================>]
2/6: running   [==================================>]
3/6: running   [==================================>]
4/6: running   [==================================>]
5/6: running   [==================================>]
6/6: running   [==================================>]
verify: Service converged
```



Docker Swarm:

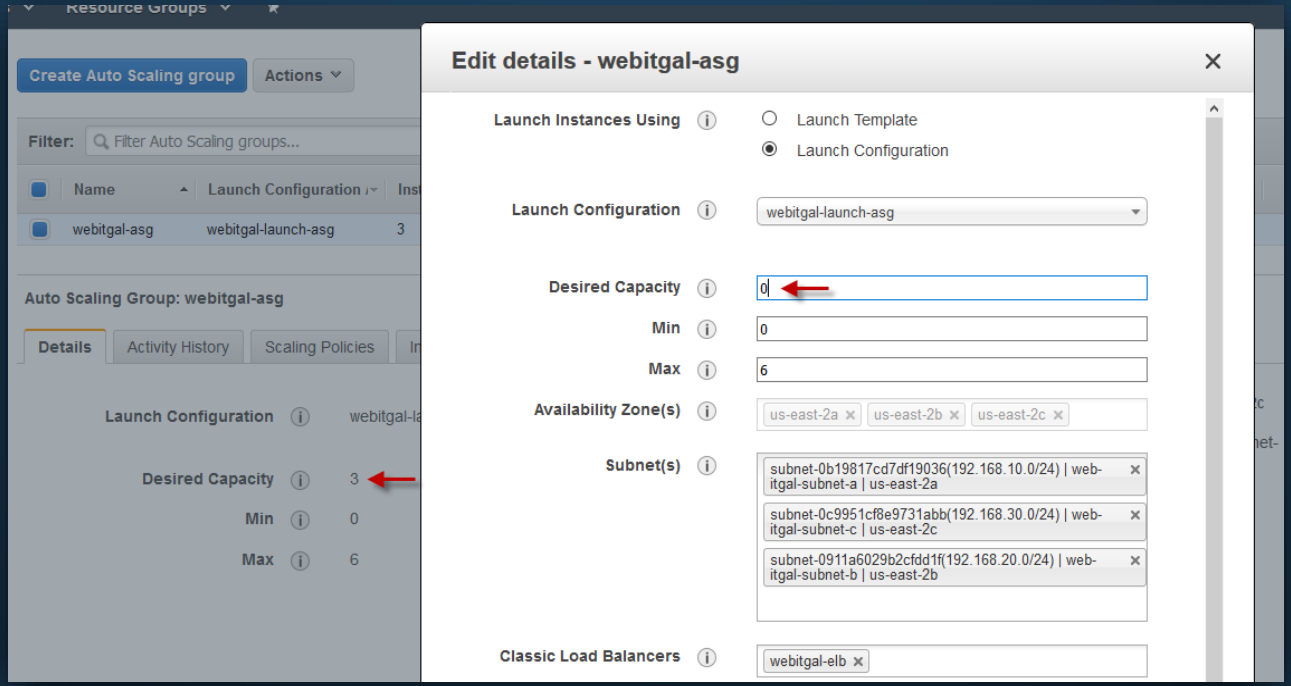
Actualización de servicio para reducir las tareas de réplicas a uno

```
root@manager:~# docker service update --replicas=1
webitgal
webitgal
overall progress: 1 out of 1 tasks
1/1: running
[==================================================>]
verify: Service converged
```



Auto Scaling Group

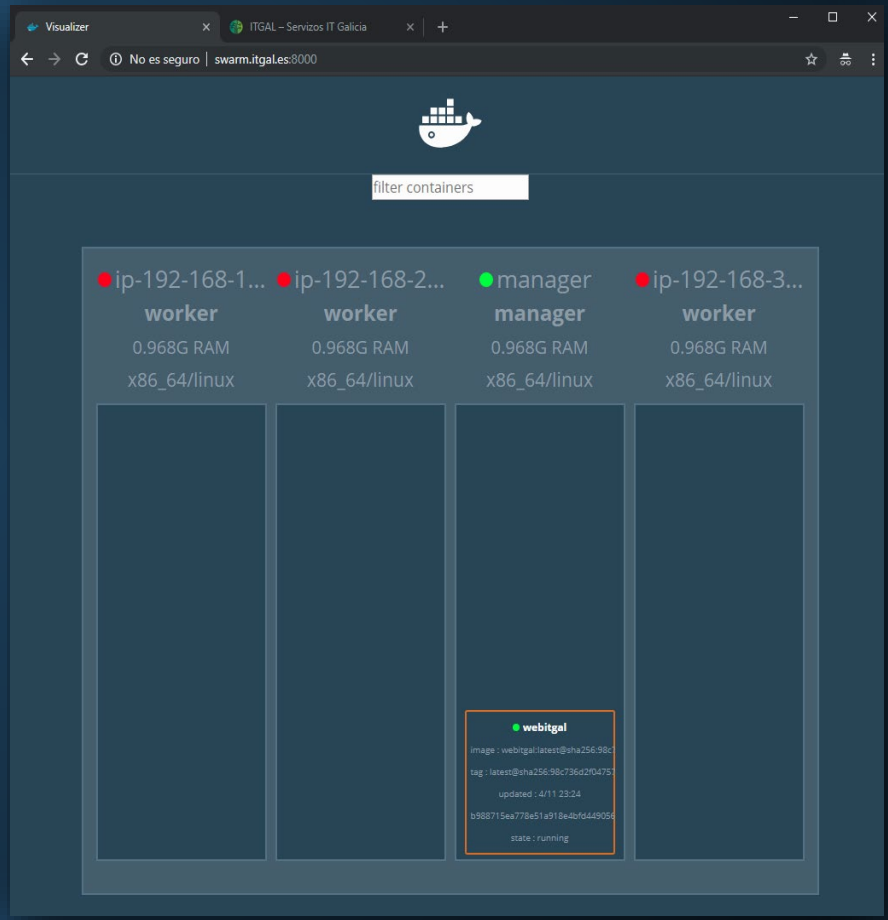
Scale-in a cero en todas las instancias desplegadas



Docker Swarm:

Después del scale-in a cero, el estado de los nodos es "Down"

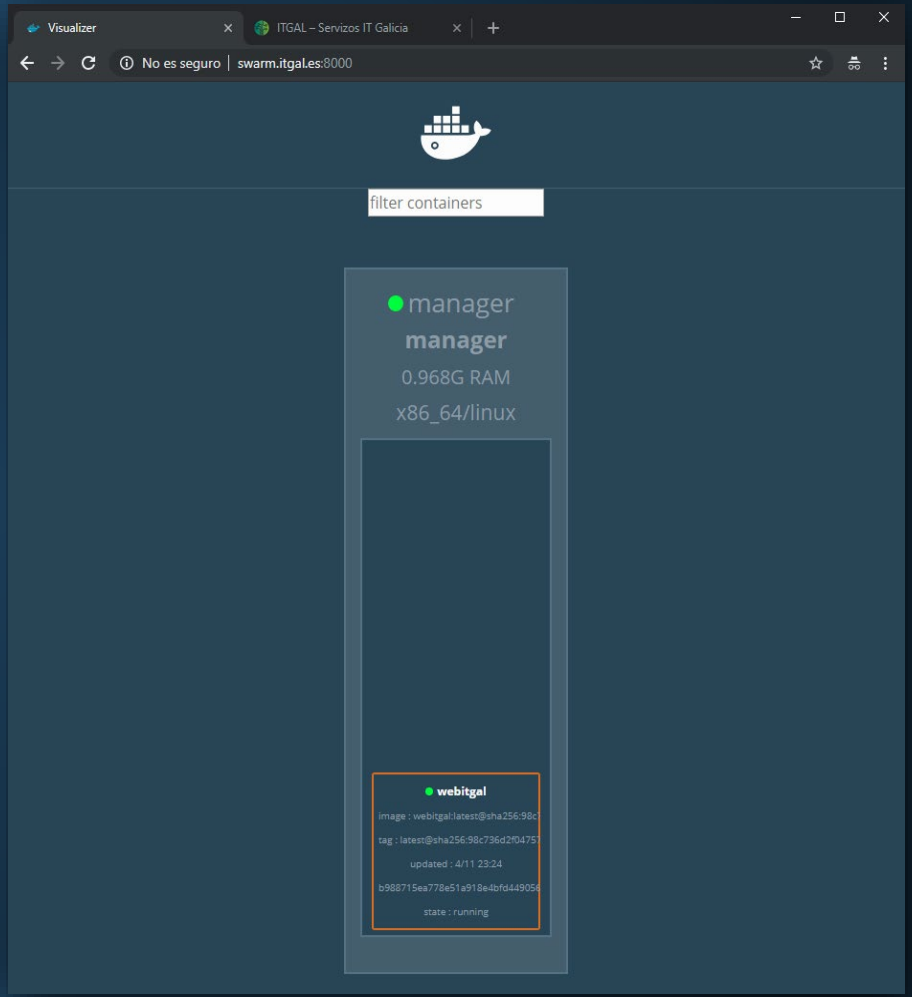
```
root@manager:~# docker node ls
ID                            HOSTNAME            STATUS
AVAILABILITY        MANAGER STATUS      ENGINE VERSION
eyxf2pqhfbprckm6db44761zj     ip-192-168-10-64    Down
Active                                  18.06.1-ce
qw84n3ev09yhc30vlmddrwpzb     ip-192-168-20-133   Down
Active                                  18.06.1-ce
togf7g1cv22rol9c9jowj1ioh     ip-192-168-30-222   Down
Active                                  18.06.1-ce
b5mm1cmv2d2jr1dgh5rlr3oqk *   manager             Ready
Active              Leader              18.06.1-ce
```



Bash Script:

Script para eliminar nodos “down” del Swarm de nodos. (Instancias en estado “terminated”)

```
#!/bin/bash
nodo_down=$(docker node ls | grep Down | sed -n '1p' |
awk '{print $1}')
while [ "$nodo_down" != "" ];
do
  nodo_down=$(docker node ls | grep Down | sed -n '1p'
| awk '{print $1}')
  docker node rm --force "$nodo_down"
done
```



$>WHOAMI

- adrianlois@zonasystem.com
- https://zonasystem.com
- @adrianlois_