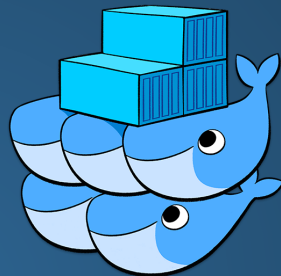


IMPLEMENTACIÓN DE  
**DOCKER SWARM**  
EN



**AMAZON WEB SERVICES**  
USANDO  
AUTO SCALING GROUPS  
Y  
ELASTIC LOAD BALANCING

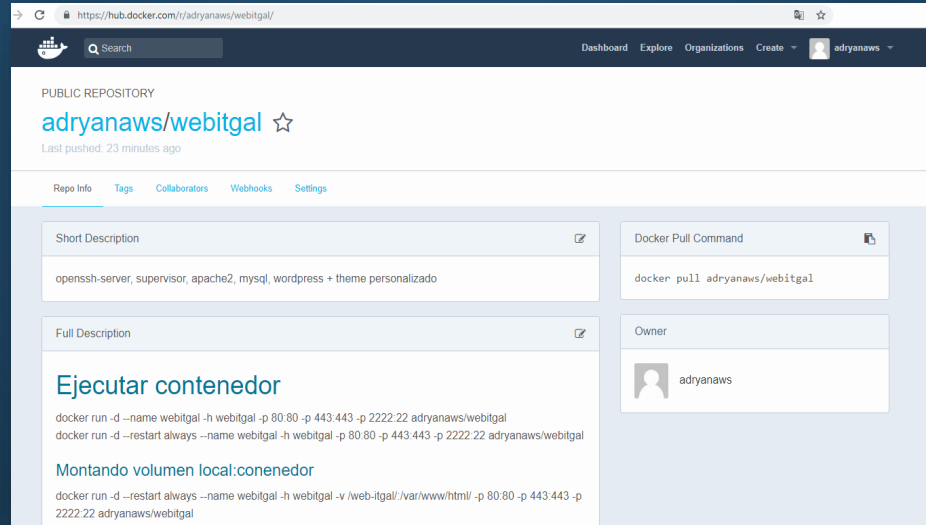
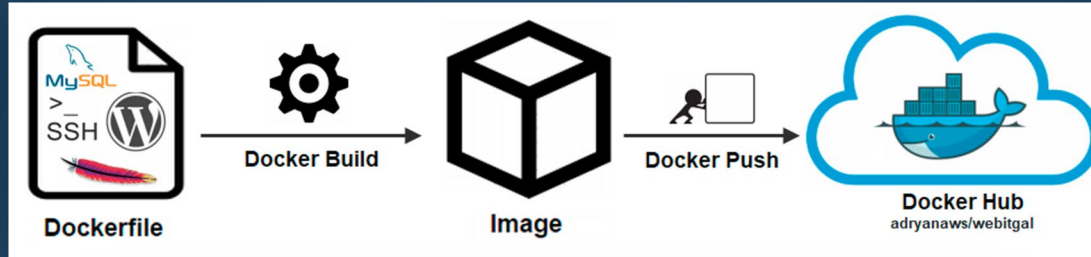
**Adrián Gómez Lois**

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

# Creando un Dockerfile

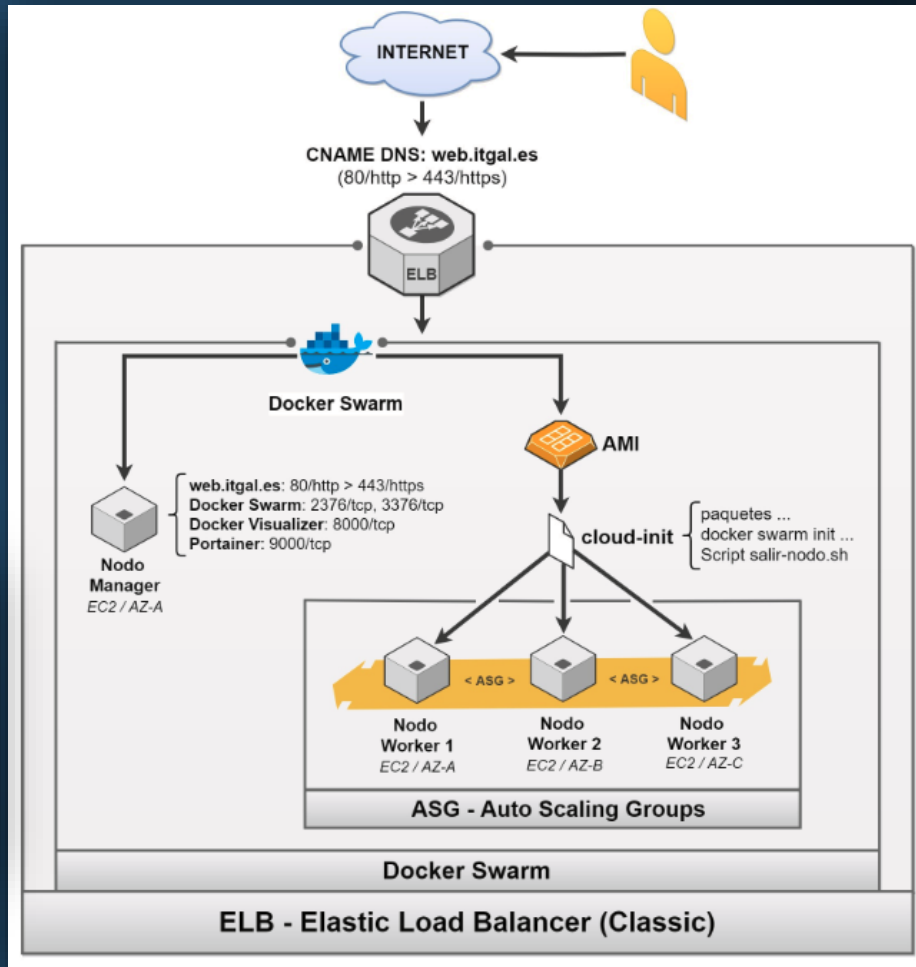
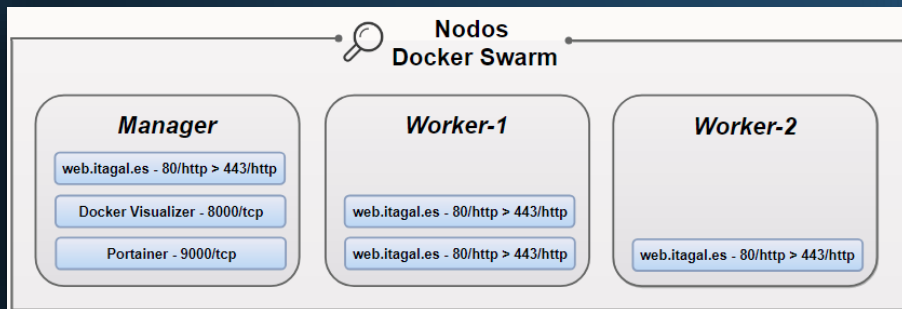
- Partirá de una imagen limpia de Debian.
- Se instalarán los paquetes: net-tools, curl, mysql, apache2, php y otros paquetes necesarios relacionados: wget, nano, curl, openssh-server y supervisor.
- Creará una BBDD Wordpress con usuario y password y le otorgará permisos.
- Copiará un contenido de una instalación previamente configurada con un theme personalizado y adaptado de Wordpress, asignará el propietario www-data al directorio.
- Creará un directorio en el contenedor y copiará en él 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 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:

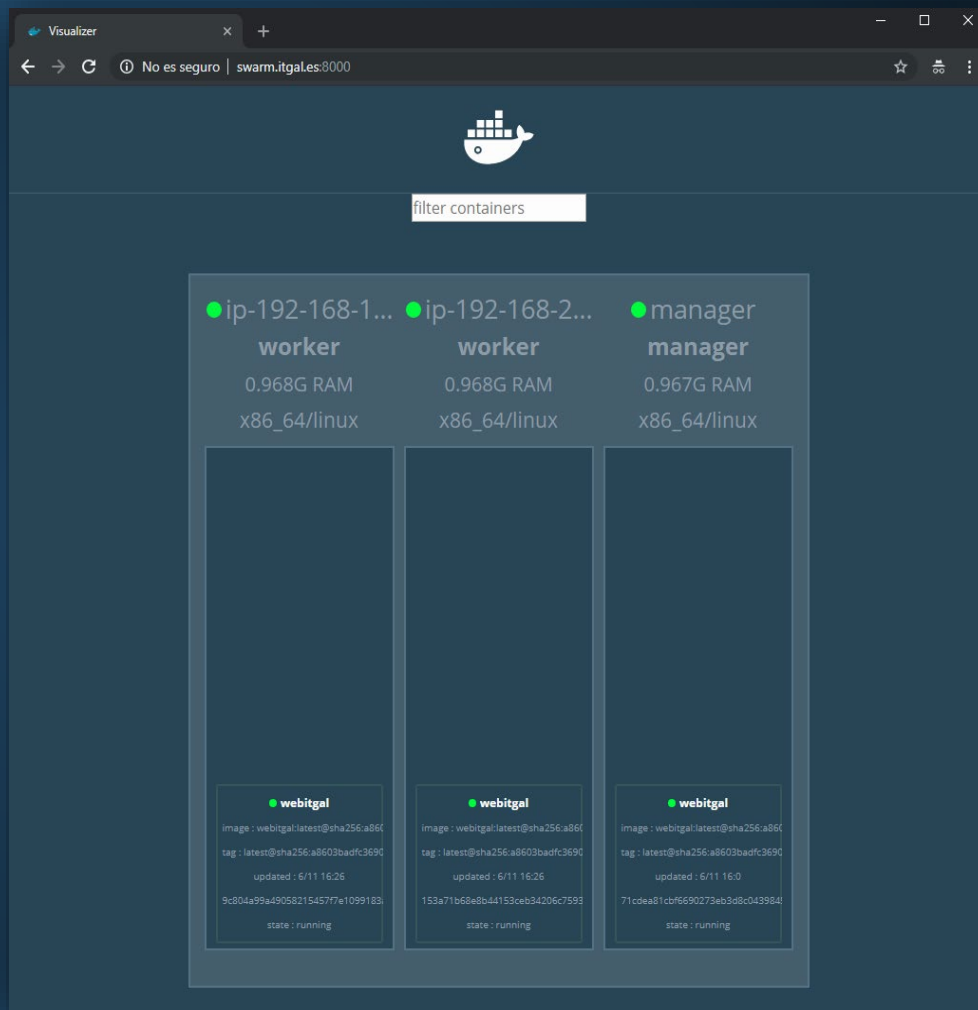
## Docker Swarm, Auto Scaling Group, Elastic Load Balancer



# Docker Swarm:

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

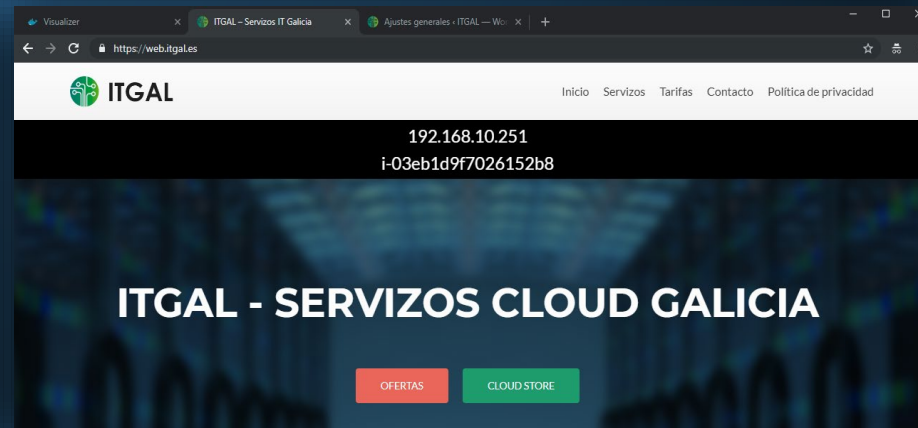
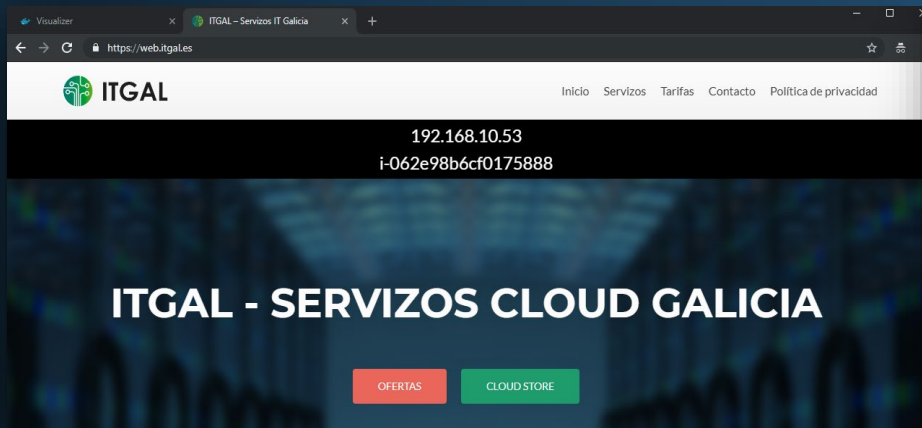
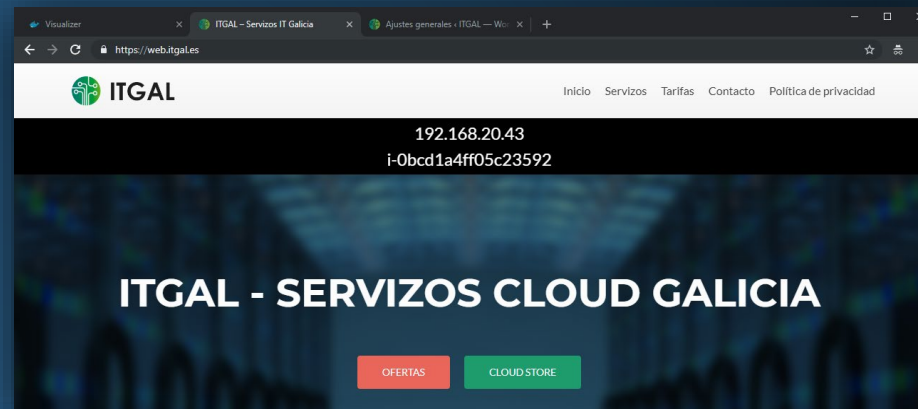
```
root@manager:~# docker service create \
> --name webitgal \
> --publish 80:80 \
> --publish 443:443 \
> --replicas 3 \
> --update-parallelism 1 \
> --update-delay 5s \
> --restart-condition on-failure \
> adryanaws/webitgal
8eoqdtcmx20mne3iepo2074px
overall progress: 3 out of 3 tasks
1/3: running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

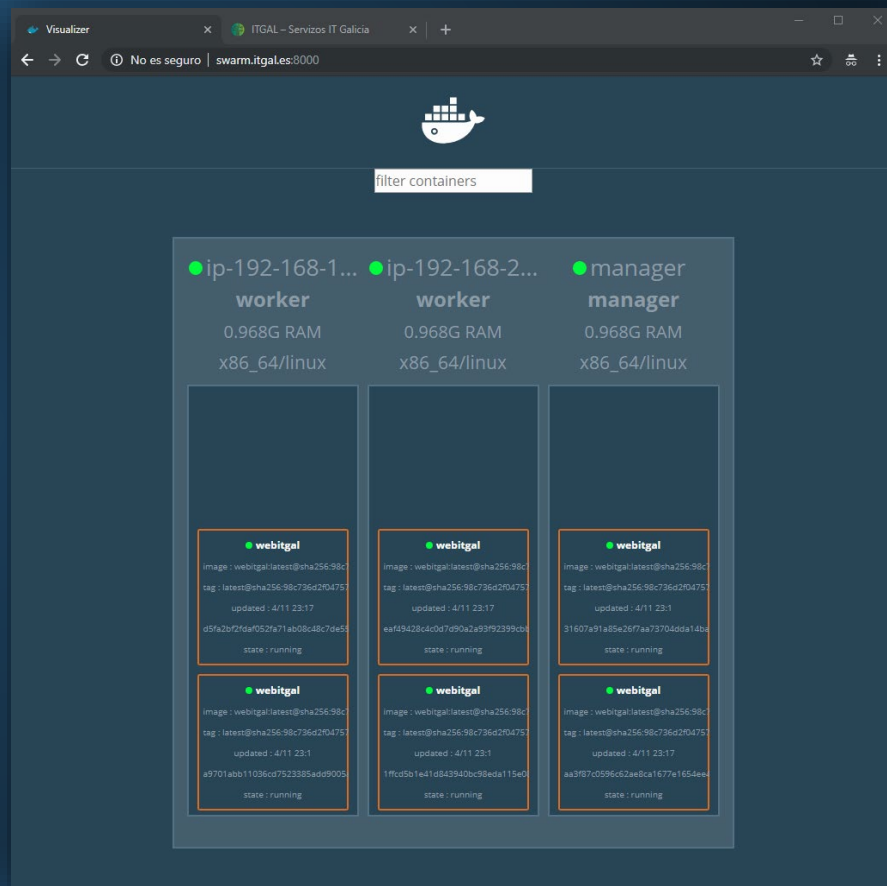




# Docker Swarm:

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

Resource Groups

Create Auto Scaling group Actions

Filter: Filter Auto Scaling groups...

Name	Launch Configuration
webitgal-asg	webitgal-launch-asg

Auto Scaling Group: webitgal-asg

Details Activity History Scaling Policies

Launch Configuration ⓘ webitgal-launch-asg

Desired Capacity ⓘ 2

Min ⓘ 0

Max ⓘ 6

Edit details - webitgal-asg

Launch Instances Using ⓘ ☐ Launch Template ☒ Launch Configuration

Launch Configuration ⓘ webitgal-launch-asg

Desired Capacity ⓘ 3

Min ⓘ 0

Max ⓘ 6

Availability Zone(s) ⓘ us-east-2a x us-east-2b x us-east-2c x

Subnet(s) ⓘ

- subnet-0b19817cd7df19036(192.168.10.0/24) | webitgal-subnet-a | us-east-2a x
- subnet-0c9951cf8e9731abb(192.168.30.0/24) | webitgal-subnet-c | us-east-2c x
- subnet-0911a6029b2cfd1f(192.168.20.0/24) | webitgal-subnet-b | us-east-2b x

Classic Load Balancers ⓘ webitgal-elb x

Target Groups ⓘ

Health Check Type ⓘ EC2

Health Check Grace Period ⓘ 100

Instance Protection ⓘ

Termination Policies ⓘ Default x

Cancel Save



# Docker Swarm:

Actualización de servicio con las mismas 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
```

The screenshot shows the Docker Swarm Visualizer interface in a web browser. The browser tabs include 'Visualizer' and 'ITGAL - Servicios IT Galicia'. The address bar shows 'swarm.itgales:8000'. The interface features a search bar labeled 'filter containers' and a grid of four nodes:

- ip-192-168-1...** (worker): 0.968G RAM, x86\_64/linux. Contains one 'webitgal' service instance.
- ip-192-168-2...** (worker): 0.968G RAM, x86\_64/linux. Contains one 'webitgal' service instance.
- manager** (manager): 0.968G RAM, x86\_64/linux. Contains two 'webitgal' service instances.
- ip-192-168-3...** (worker): 0.968G RAM, x86\_64/linux. Contains two 'webitgal' service instances.

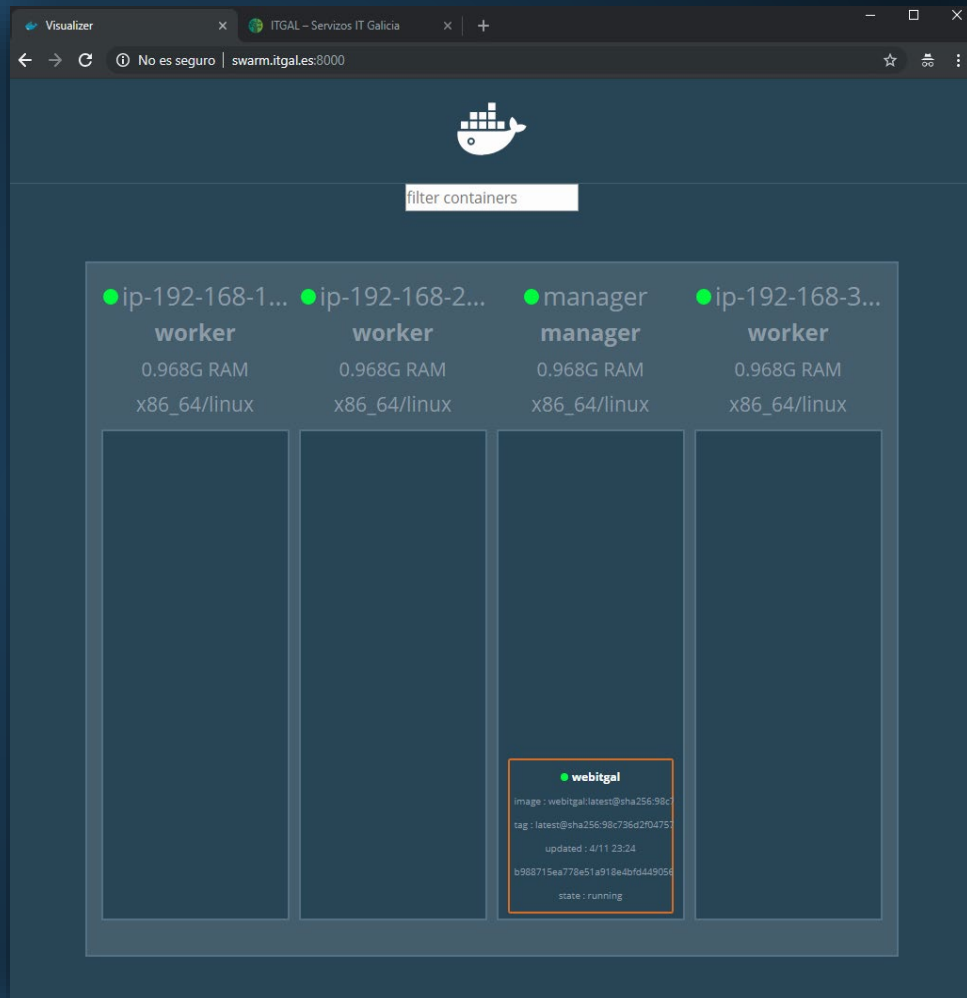
Each 'webitgal' service instance card displays the following details:

- Image: webitgal:latest@sha256:98c...
- Tag: latest@sha256:98c736d2f0475f...
- Updated: 4/11 23:24
- State: running

# 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

The screenshot displays the AWS Management Console interface for an Auto Scaling Group named 'webitgal-asg'. The left sidebar shows the 'Resource Groups' section with a 'Create Auto Scaling group' button and a table listing the group. The main panel shows the 'Edit details - webitgal-asg' configuration window.

**Left Sidebar:**

- Buttons: 'Create Auto Scaling group', 'Actions'.
- Filter: 'Filter Auto Scaling groups...'
- Table:

Name	Launch Configuration	Instances
webitgal-asg	webitgal-launch-asg	3

**Auto Scaling Group: webitgal-asg**

Details | Activity History | Scaling Policies | In

Launch Configuration ⓘ webitgal-la

Desired Capacity ⓘ 3

Min ⓘ 0

Max ⓘ 6

**Edit details - webitgal-asg**

Launch Instances Using ⓘ

- ☐ Launch Template
- ☒ Launch Configuration

Launch Configuration ⓘ webitgal-launch-asg

Desired Capacity ⓘ 0

Min ⓘ 0

Max ⓘ 6

Availability Zone(s) ⓘ us-east-2a x us-east-2b x us-east-2c x

Subnet(s) ⓘ

- subnet-0b19817cd7df19036(192.168.10.0/24) | webitgal-subnet-a | us-east-2a x
- subnet-0c9951cf8e9731abb(192.168.30.0/24) | webitgal-subnet-c | us-east-2c x
- subnet-0911a6029b2cddd1f(192.168.20.0/24) | webitgal-subnet-b | us-east-2b x

Classic Load Balancers ⓘ webitgal-elb x

# Docker Swarm:

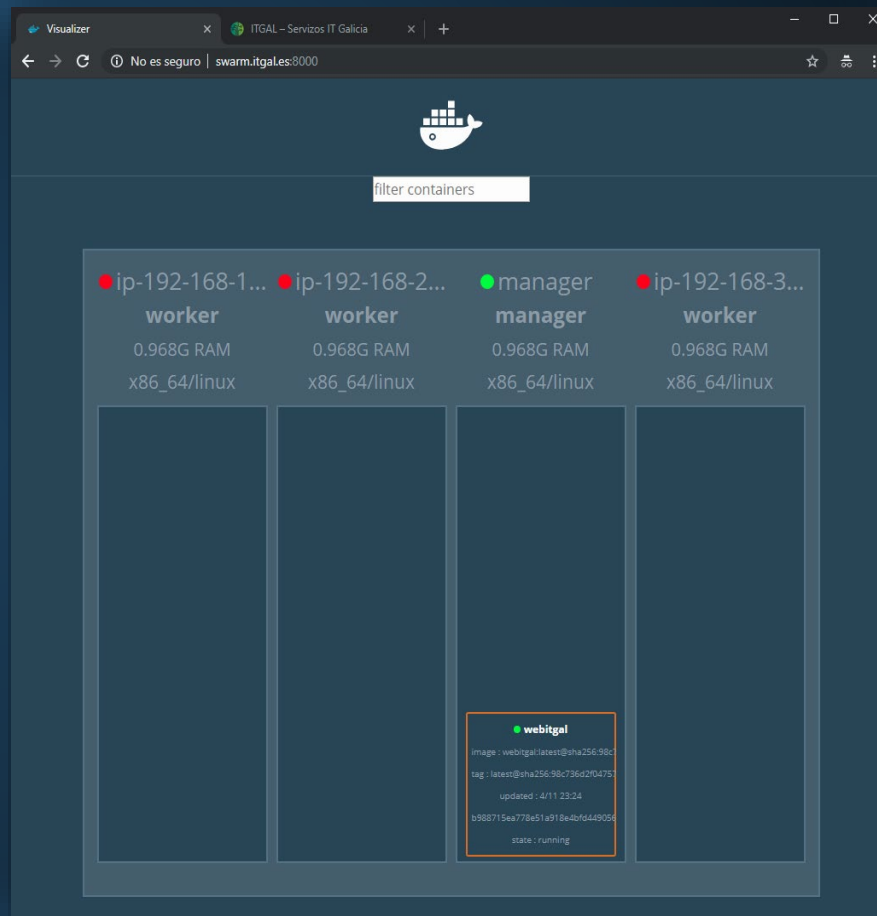
Después del scale-in a cero, el estado de los nodos es "Down"

```
root@manager:~# docker node ls
```

ID	HOSTNAME	STATUS
eyxf2pqhfbprckm6db44761zj	ip-192-168-10-64	Down
qw84n3ev09yhc30vlmddrwpzb	ip-192-168-20-133	Down
togf7g1cv22rol9c9jowj1ioh	ip-192-168-30-222	Down
b5mm1cmv2d2jr1dgh5r1r3oqk *	manager	Ready

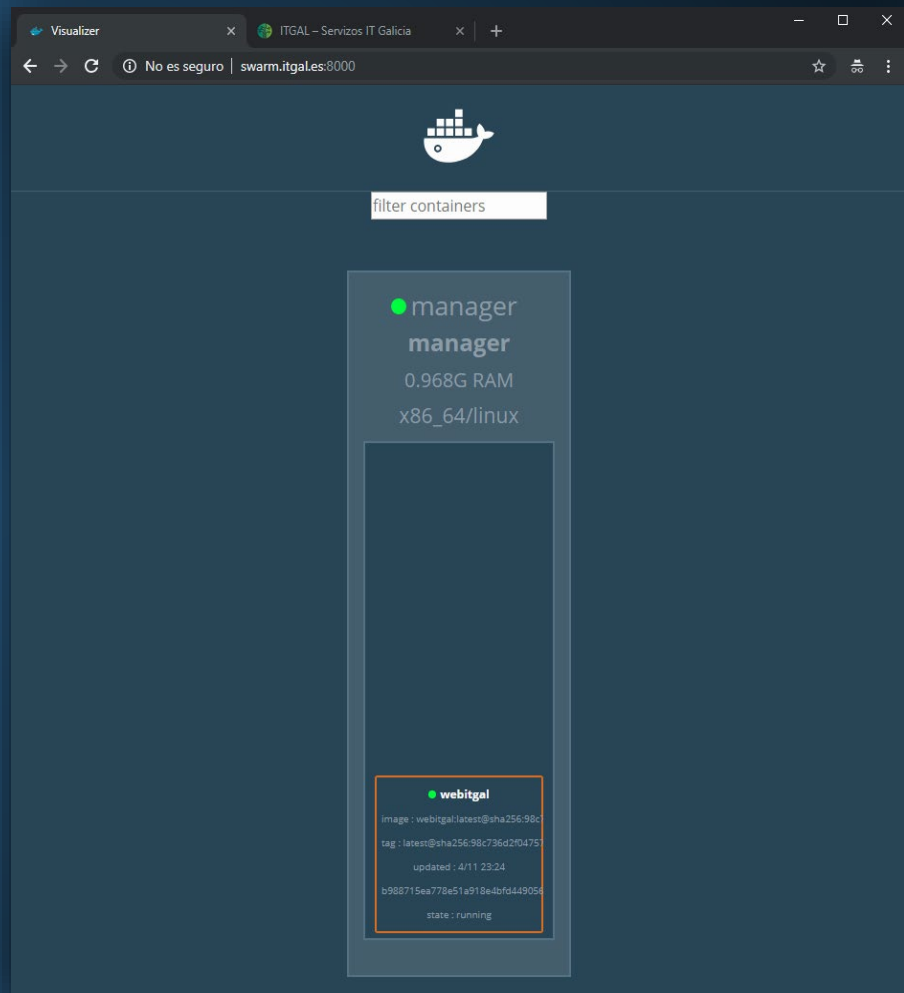
AVAILABILITY	MANAGER	STATUS	ENGINE	VERSION
Active			18.06.1-ce	
Active			18.06.1-ce	
Active			18.06.1-ce	
Active	Leader		18.06.1-ce	



## 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](mailto:adrianlois@zonasystem.com)
- <https://zonasystem.com>
- [@adrianlois\\_](https://twitter.com/adrianlois)

