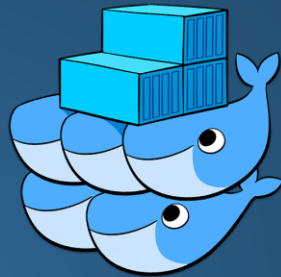


IMPLEMENTACIÓN DE  
**DOCKER SWARM**  
EN



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

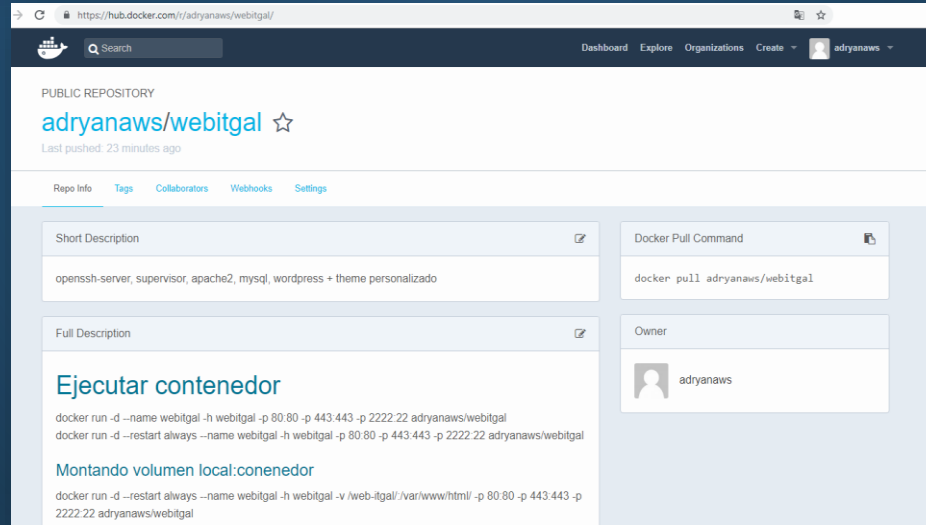
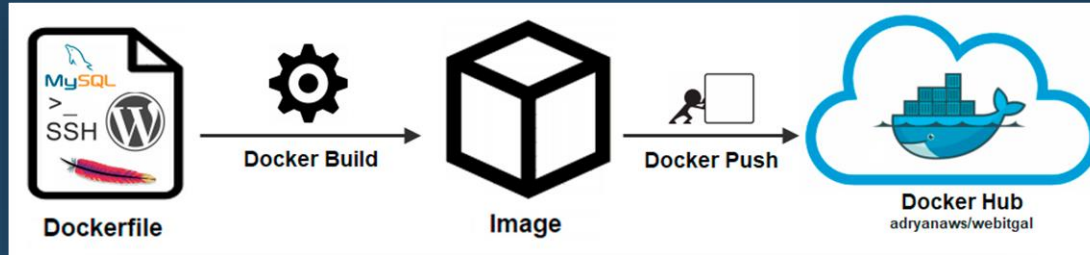
**Adrián Gómez Lois**

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

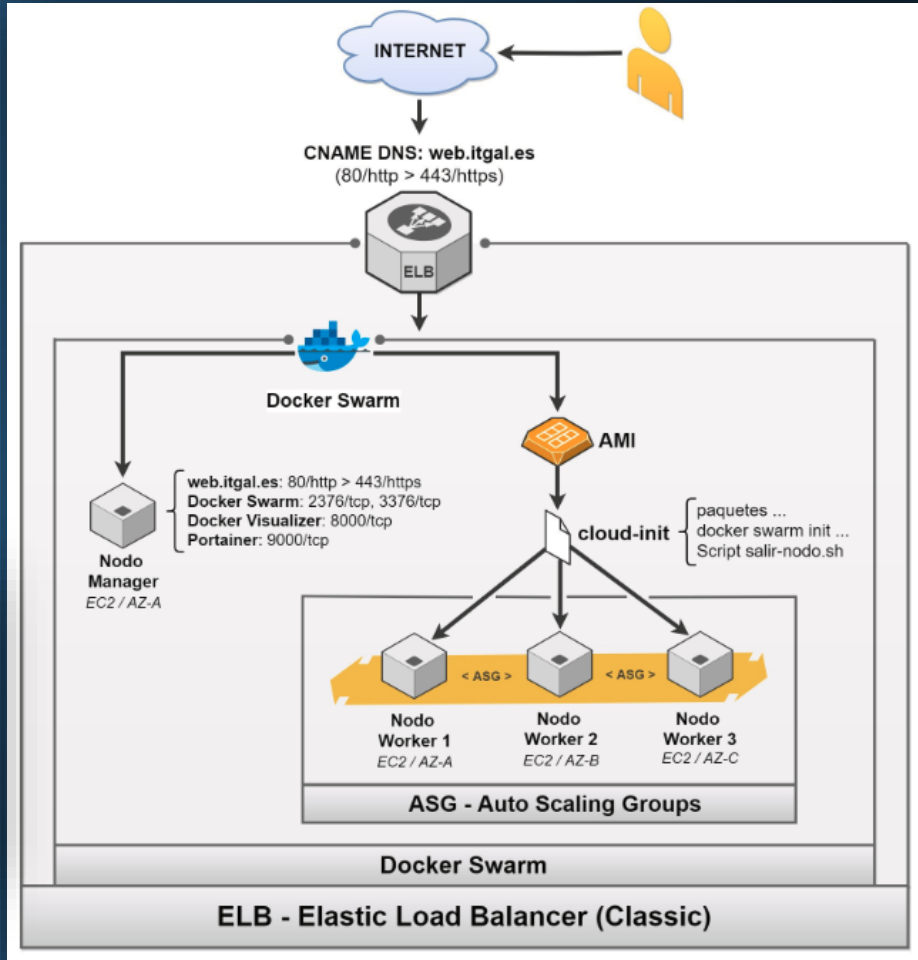
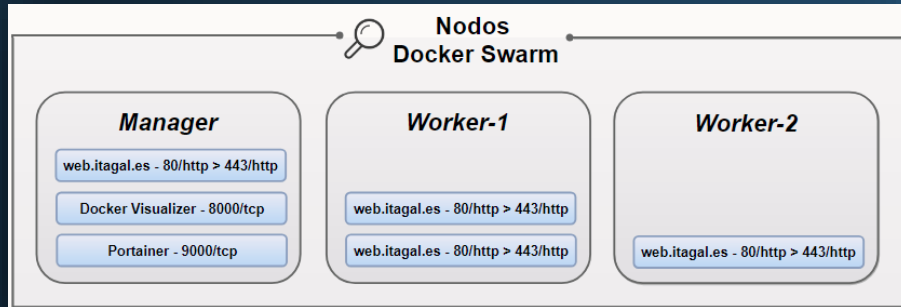
# Creando un Dockerfile

- Partirá de una imagen limpia de Debian.
- Instalará 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 para la conexión SSH al contenedor.
- Establecerá un banner motd personalizado.
- Expondrá públicamente los puertos para SSH/22, HTTP/80 y HTTPS/443.
- Establecerá como entrypoint (CMD) del contenedor la ejecución de supervisord.

# Subir imagen a Docker Hub



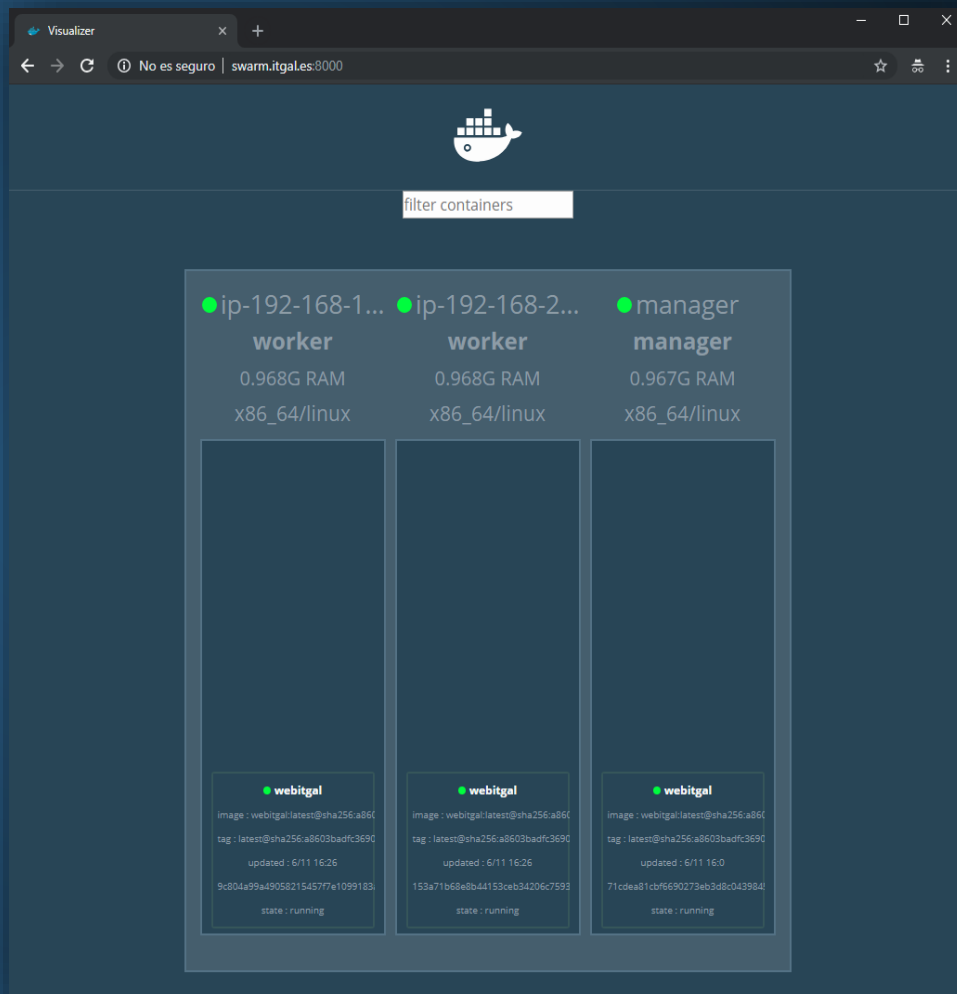
# Diagrama AWS: Docker Swarm, Auto Scaling Group, Elastic Load Balancer



# 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:

Balanceador de carga entre instancias EC2 gestionadas por ASG e instancia fija correspondiente al nodo manager

The screenshot shows the AWS Management Console interface for the Elastic Load Balancing service. The left sidebar contains navigation links for various AWS services, including Lifecycle Manager, NETWORK & SECURITY, Security Groups, Elastic IPs, Placement Groups, Key Pairs, Network Interfaces, LOAD BALANCING, Load Balancers, Target Groups, AUTO SCALING, Launch Configurations, Auto Scaling Groups, SYSTEMS MANAGER SERVICES, Run Command, State Manager, Configuration Compliance, Automations, Patch Compliance, Patch Baselines, and SYSTEMS MANAGER SHARED RESOURCES.

The main content area displays the 'Create Load Balancer' button and a search bar. Below the search bar is a table of load balancers. The table has columns for Name, DNS name, State, VPC ID, Availability Zones, and Type. The first row shows a load balancer named 'webitgal-elb' with a DNS name of 'webitgal-elb-1790521091.us-...', VPC ID of 'vpc-0d1dd38b31455299f', and availability zones of 'us-east-2a, us-east-2b, ...'. The type is 'classic'.

Below the table, the 'Load balancer: webitgal-elb' section is shown. It includes tabs for Description, Instances, Health Check, Listeners, Monitoring, Tags, and Migration. The 'Instances' tab is selected, showing a table of EC2 instances. The table has columns for Instance ID, Name, Availability Zone, and Status. The instances are:

Instance ID	Name	Availability Zone	Status
i-03eb1d9f026152b8	manager	us-east-2a	InService ⓘ
i-099cfa1f37b387fd9	worker-asg	us-east-2a	InService ⓘ
i-091d9052f05d11941	worker-asg	us-east-2c	InService ⓘ
i-0e75f36d199d19785	worker-asg	us-east-2b	InService ⓘ

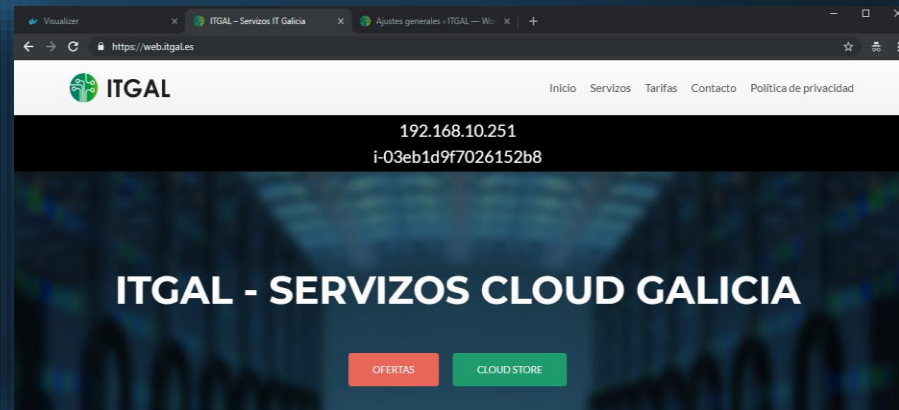
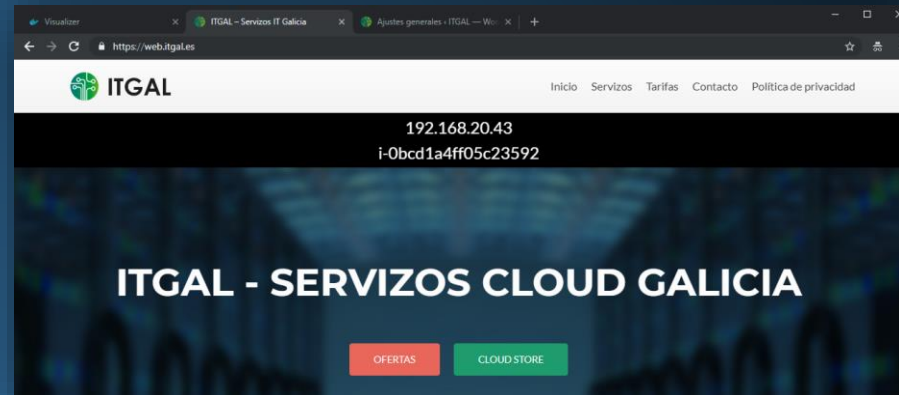
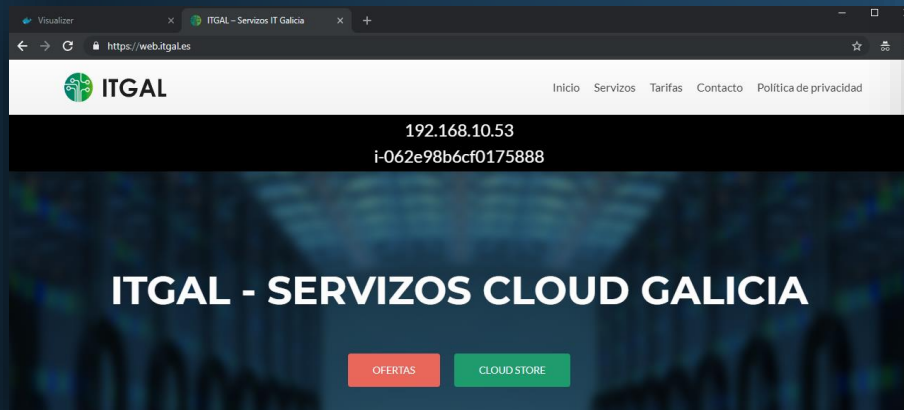
Below the instances table, the 'Edit Availability Zones' section is shown. It includes a table of availability zones. The table has columns for Availability Zone, Subnet ID, Subnet CIDR, Instance Count, and Healthy?. The availability zones are:

Availability Zone	Subnet ID	Subnet CIDR	Instance Count	Healthy?
us-east-2a	subnet-0b19817cd7df19036	192.168.10.0/24	2	Yes
us-east-2b	subnet-0911a6029b2cfd1f	192.168.20.0/24	1	Yes
us-east-2c	subnet-0c9951cfe9731abb	192.168.30.0/24	1	Yes

# 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>
```

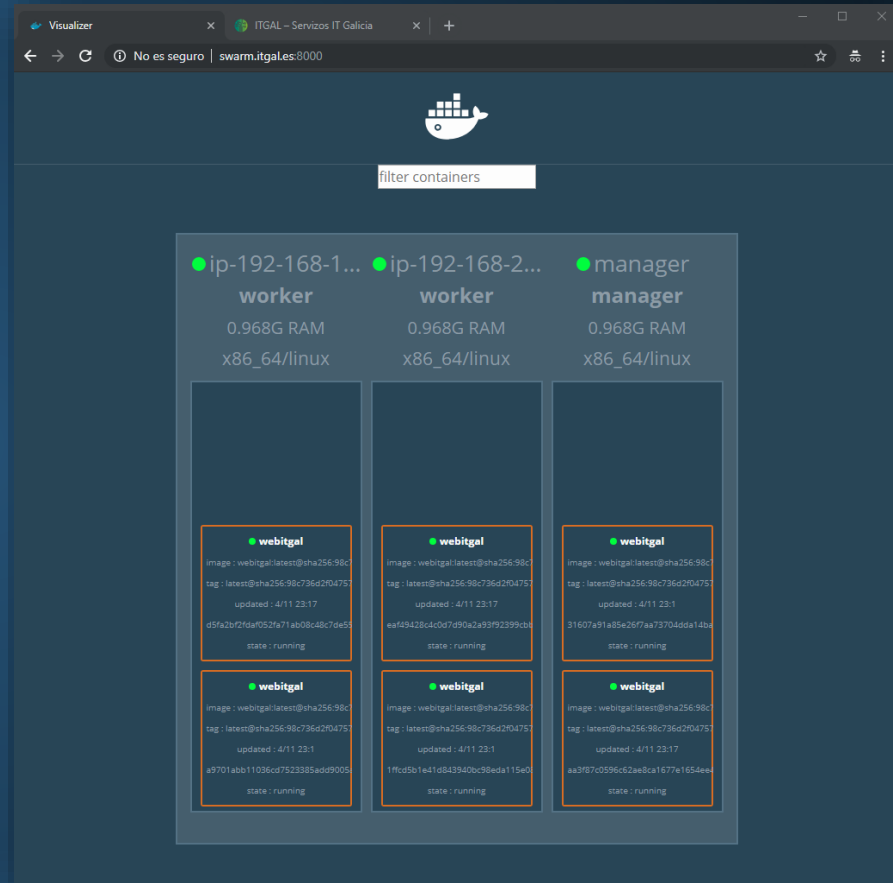




# Docker Swarm:

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

The image shows the AWS Auto Scaling console. On the left, a list of Auto Scaling Groups is visible, with 'webitgal-asg' selected. A red arrow points to its 'Desired Capacity' of 2. The main panel displays the 'Edit details - webitgal-asg' dialog. In this dialog, the 'Launch Configuration' is set to 'webitgal-launch-asg'. The 'Desired Capacity' is set to 3, with a red arrow pointing to the input field. The 'Min' capacity is 0 and the 'Max' capacity is 6. The 'Availability Zone(s)' are set to 'us-east-2a', 'us-east-2b', and 'us-east-2c'. The 'Subnet(s)' are set to 'subnet-0b19817cd7df19036(192.168.10.0/24) | webitgal-subnet-a | us-east-2a', 'subnet-0c9951cf8e9731abb(192.168.30.0/24) | webitgal-subnet-c | us-east-2c', and 'subnet-0911a6029b2cdd1ff(192.168.20.0/24) | webitgal-subnet-b | us-east-2b'. The 'Classic Load Balancers' are set to 'webitgal-elb'. The 'Health Check Type' is 'EC2', the 'Health Check Grace Period' is 100, and the 'Termination Policies' are 'Default'. The 'Launch Instances Using' option is set to 'Launch Configuration'.

**Edit details - webitgal-asg**

Launch Instances Using *i* ☐ Launch Template ☒ Launch Configuration

Launch Configuration *i* webitgal-launch-asg

Desired Capacity *i* 3 *←*

Min *i* 0

Max *i* 6

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

Subnet(s) *i*

- 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-0911a6029b2cdd1ff(192.168.20.0/24) | webitgal-subnet-b | us-east-2b x

Classic Load Balancers *i* webitgal-elb x

Target Groups *i*

Health Check Type *i* EC2

Health Check Grace Period *i* 100

Instance Protection *i*

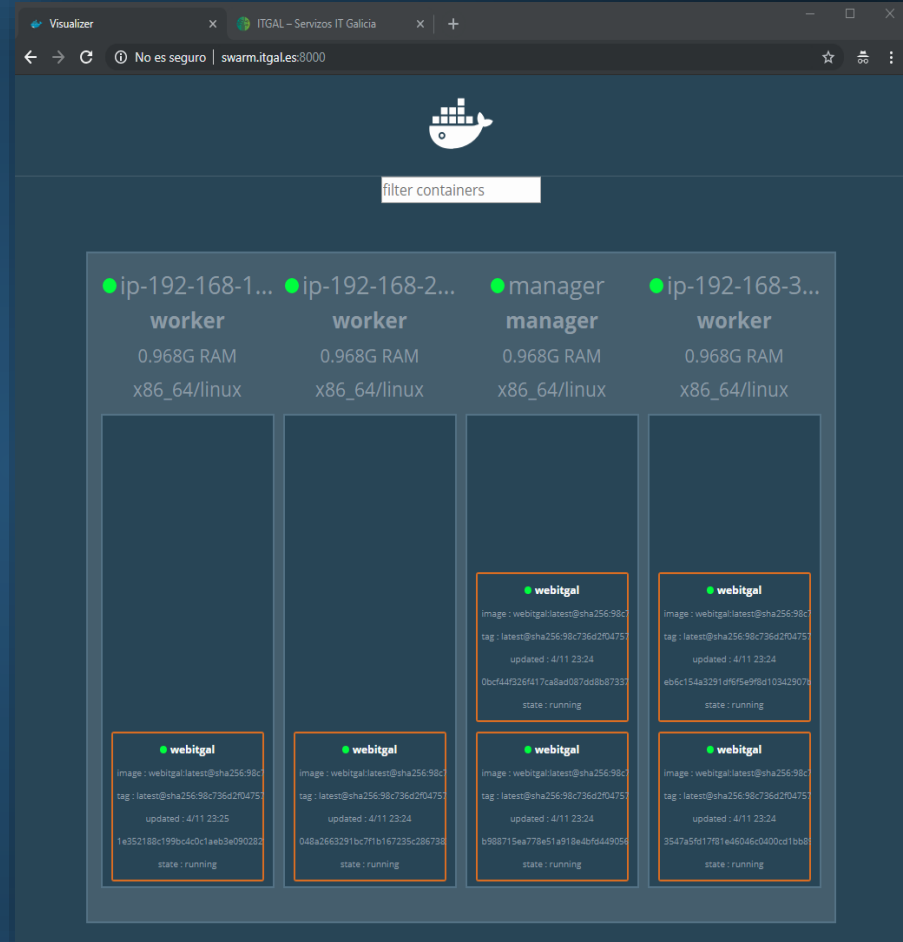
Termination Policies *i* Default x

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

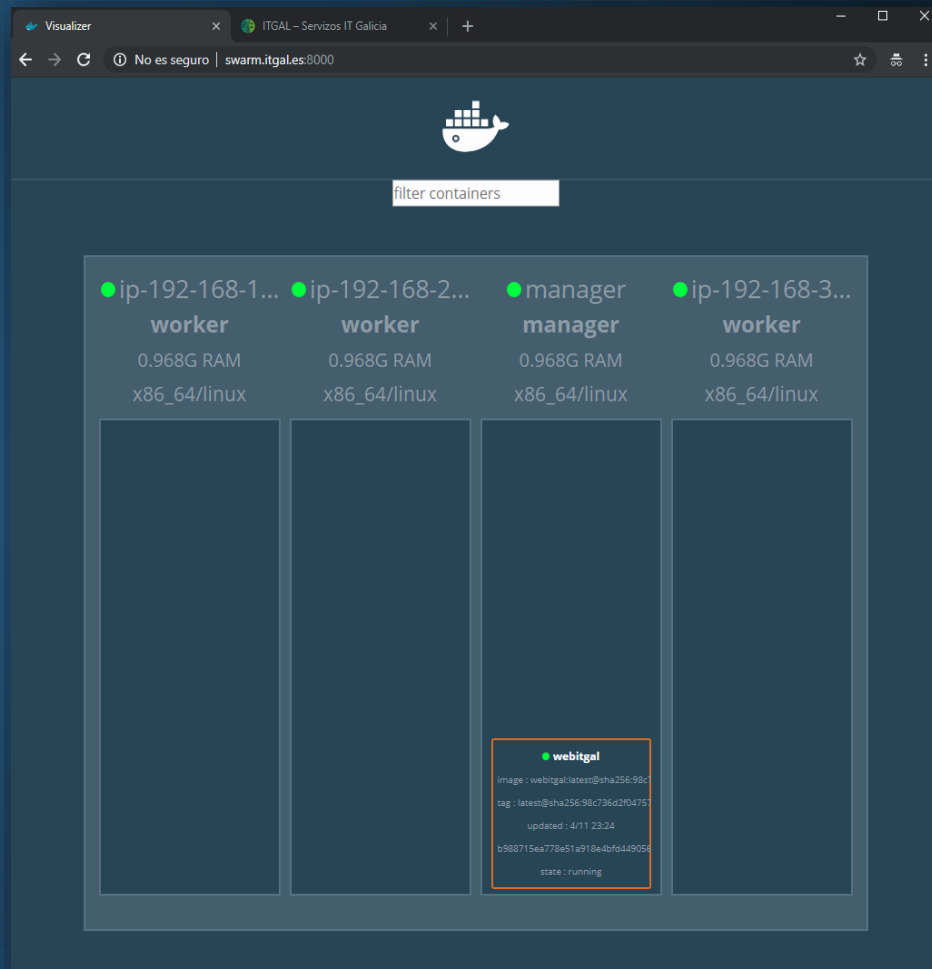
```
root@manager:~# docker service update --force webitgal
webitgal
overall progress: 6 out of 6 tasks
1/6: running [=====>]
2/6: running [=====>]
3/6: running [=====>]
4/6: running [=====>]
5/6: running [=====>]
6/6: running [=====>]
verify: Service converged
```



# 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 image shows the AWS Management Console interface for an Auto Scaling Group named 'webitgal-asg'. On the left, the 'Details' tab is active, showing the 'Desired Capacity' as 3, with a red arrow pointing to it. The 'Launch Configuration' is 'webitgal-launch-asg'. On the right, the 'Edit details - webitgal-asg' modal is open. In this modal, the 'Launch Instances Using' section has 'Launch Configuration' selected. The 'Launch Configuration' dropdown is set to 'webitgal-launch-asg'. The 'Desired Capacity' input field is highlighted with a red arrow and contains the value '0'. The 'Min' input field also contains '0', and the 'Max' input field contains '6'. The 'Availability Zone(s)' section shows three zones: 'us-east-2a', 'us-east-2b', and 'us-east-2c'. The 'Subnet(s)' section lists three subnets: 'subnet-0b19817cd7df19036(192.168.10.0/24) | webitgal-subnet-a | us-east-2a', 'subnet-0c9951cf8e9731abb(192.168.30.0/24) | webitgal-subnet-c | us-east-2c', and 'subnet-0911a6029b2cddd1f(192.168.20.0/24) | webitgal-subnet-b | us-east-2b'. The 'Classic Load Balancers' section shows 'webitgal-elb'.

**Create Auto Scaling group** **Actions**

Filter:

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

**Auto Scaling Group: webitgal-asg**

**Details** Activity History Scaling Policies

Launch Configuration *i* webitgal-launch-asg

Desired Capacity *i* 3 *i*

Min *i* 0

Max *i* 6

**Edit details - webitgal-asg**

Launch Instances Using *i*

☐ Launch Template

☒ Launch Configuration

Launch Configuration *i* webitgal-launch-asg

Desired Capacity *i* 0 *i*

Min *i* 0

Max *i* 6

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

Subnet(s) *i*

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

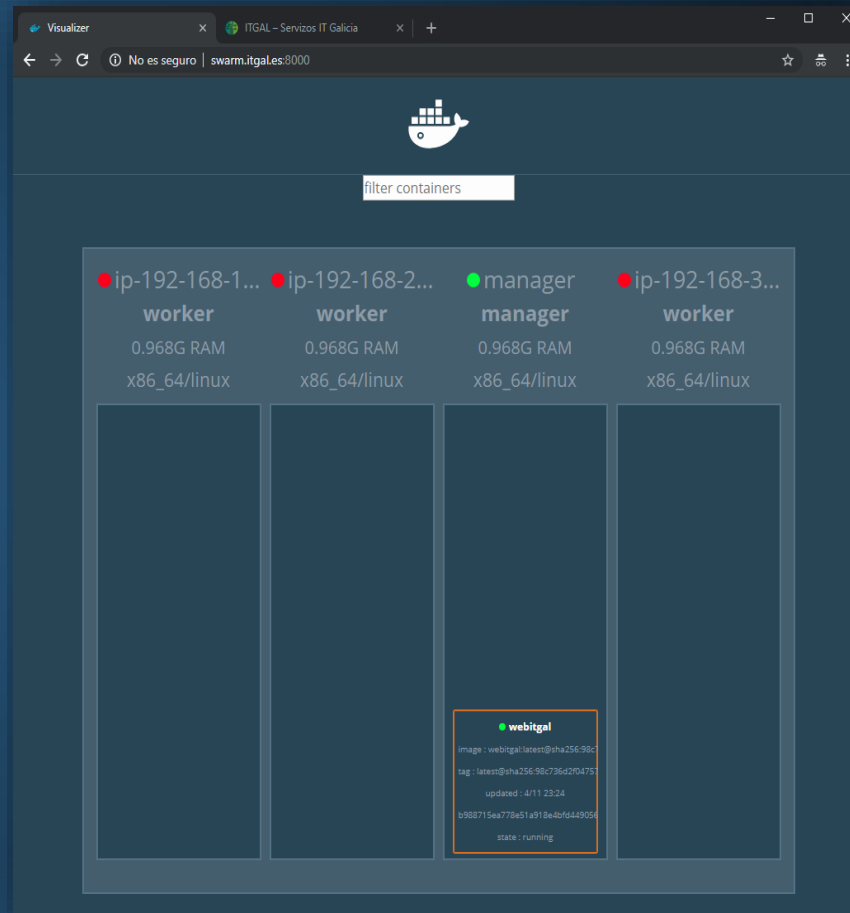
Classic Load Balancers *i* webitgal-elb x

# Docker Swarm:

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

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

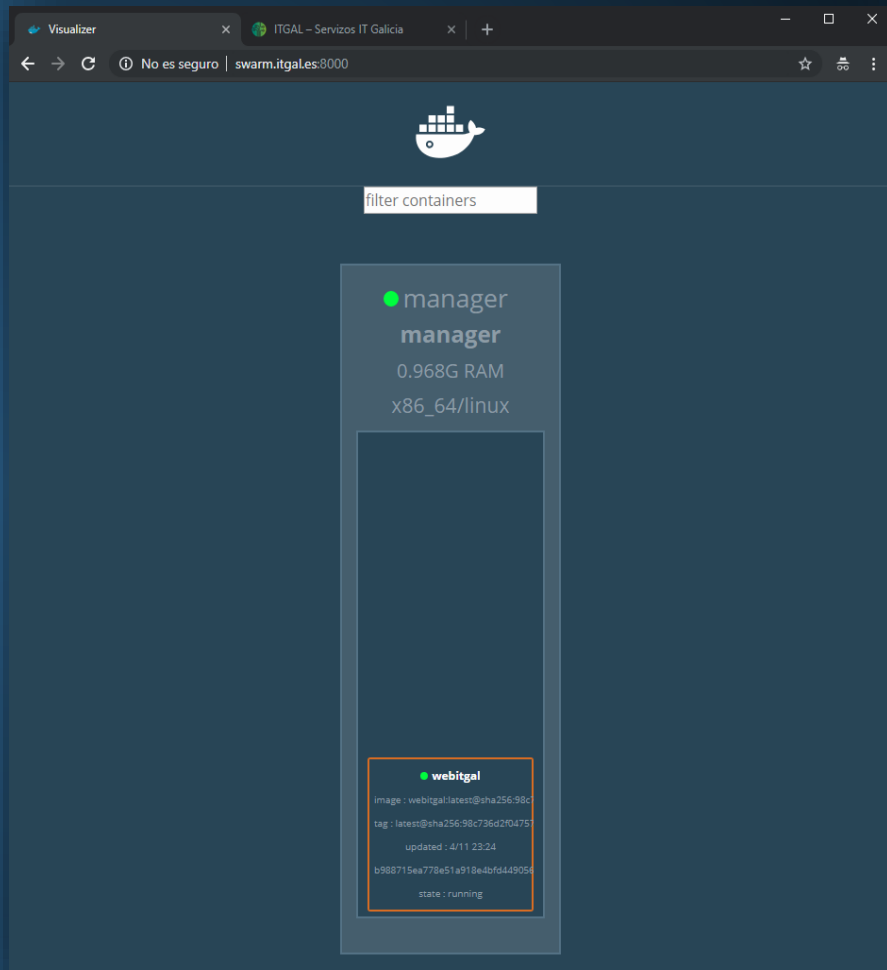
ID	HOSTNAME	STATUS
AVAILABILITY	MANAGER	STATUS
ENGINE	VERSION	
eyxf2pqhfbprckm6db44761zj	ip-192-168-10-64	Down
Active		18.06.1-ce
qw84n3ev09yhc30vlmddrwpzb	ip-192-168-20-133	Down
Active		18.06.1-ce
togf7g1cv22rol9c9jowj1ioh	ip-192-168-30-222	Down
Active		18.06.1-ce
b5mm1cmv2d2jr1dgh5r1r3oqk *	manager	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](mailto:adrianlois@zonasystem.com)
- <https://zonasystem.com>
- @adrianlois\_

