# Auto Escalado en AWS

## Auto Scaling Group y Clúster Kubernetes

Carlos Jesús Sánchez Ortega
I.E.S Gonzalo Nazareno
Dos Hermanas - Sevilla

1. INTRODUCCIÓN A AMAZON WEB SERVICES

2. AUTO SCALING GROUPS

3. KOPS Y CLUSTER AUTO-SCALER

# INTRODUCCIÓN A AMAZON WEB SERVICES

SERVICIOS DE RED Y COMPUTACIÓN

# NETWORKING

VPC ➜ Nuestra red “local” dentro de nuestro “CPD

Subnet ➜ Subred asociada a una VPC con un “CIDR Block”

Routing Table ➜ Enrutar el tráfico de una subnet determinada

Int. Gateway ➜ Asociada a la VPC y a una Routing Table (Subnet Pública)

Elastic IP ➜ Se le denomina a las IP’s Públicas fijas

Nat Gateway ➜ Dentro de una Subnet Pública y con el Allocation ID

NACL ➜ Lista de control de acceso a la VPC

# COMPUTING

| | | |
|---|---|---|
| Security Groups | ➜ | Un firewall lógico para controlar el acceso |
| Instancias EC2 | ➜ | Es como se denominan las máquinas virtuales |
| Elastic Block Store | ➜ | Administrar y crear volúmenes en AWS |

# AUTO SCALING GROUPS

FIJAR UNA EC2 COMO PLANTILLA Y ESCALAR INSTANCIAS HORIZONTALMENTE

# PREPARAR AMI PERSONALIZADA

Script

```
#!/bin/bash
yum update -y
yum install -y httpd24 php56
service httpd start
chkconfig httpd on
```

Levantar Instancia EC2

```
aws ec2 run-instances
     --image-id ami-ca0135b3 \
     --count 1 \
     --instance-type t2.micro \
     --key-name aws-charlie \
     --subnet-id subnet-88c87aee \
     --security-group-ids sg-0d035270 \
     --user-data file://script.txt
```

# LAUNCH CONFIGURATION

Es básicamente una plantilla base de las nuevas instancias que serán lanzadas dentro del Auto Scaling Group

## CREACIÓN DE LA AMI

```
aws ec2 create-image \
     --instance-id i-0bdf383ea378fae47 \
     --name "Apache Server" \
     --no-reboot
{
    "ImageId": "ami-1414176d"
}
```

## CREACIÓN DEL LAUNCH CONFIGURATION

```
aws autoscaling create-launch-configuration \
    --launch-configuration-name launch-web \
    --security-groups sg-0d035270 \
    --image-id ami-1414176d \
    --instance-type t2.micro \
    --key-name aws-charlie
```

# AUTO SCALING GROUP

## CREACIÓN DEL ASG

```
aws autoscaling create-auto-scaling-group \
     --auto-scaling-group-name "web-servers" \
     --launch-configuration-name "launch-web" \
     --min-size 1 \
     --max-size 2 \
     --desired-capacity 1 \
     --vpc-zone-identifier "subnet-88c87aee,subnet-2364e76b" \
     --tags "ResourceId=web-servers,ResourceType=auto-scaling-group,\
     Key=Name,Value=web-server,PropagateAtLaunch=true"
```

# ACTUALIZAR EL AUTO SCALING GROUP

¡QUIERO UNA NUEVA VERSIÓN PHP EN MIS INSTANCIAS!

Pasos:

1. Conectar a la instancia inicial y realizar dicho cambio
2. Generar una nueva AMI a partir de la instancia anterior
3. Modificar el “Launch Configuration” para que utilice la nueva AMI

# ELASTIC LOAD BALANCER (ELB)

## 1. Crear un Security Group

```
aws ec2 create-security-group \
     --group-name "SG-ELB" \
     --description "SG para nuestro ELB" \
     --vpc-id "vpc-fd6f139b“
{
     "GroupId": "sg-15dd8e68"
}
```

## 2. Habilitar el puerto 80 TCP al ELB

```
aws ec2 authorize-security-group-ingress \
     --group-id "sg-15dd8e68" \
     --protocol tcp --port 80 \
     --cidr 0.0.0.0/0
```

## 3. Crear Balanceador de Carga

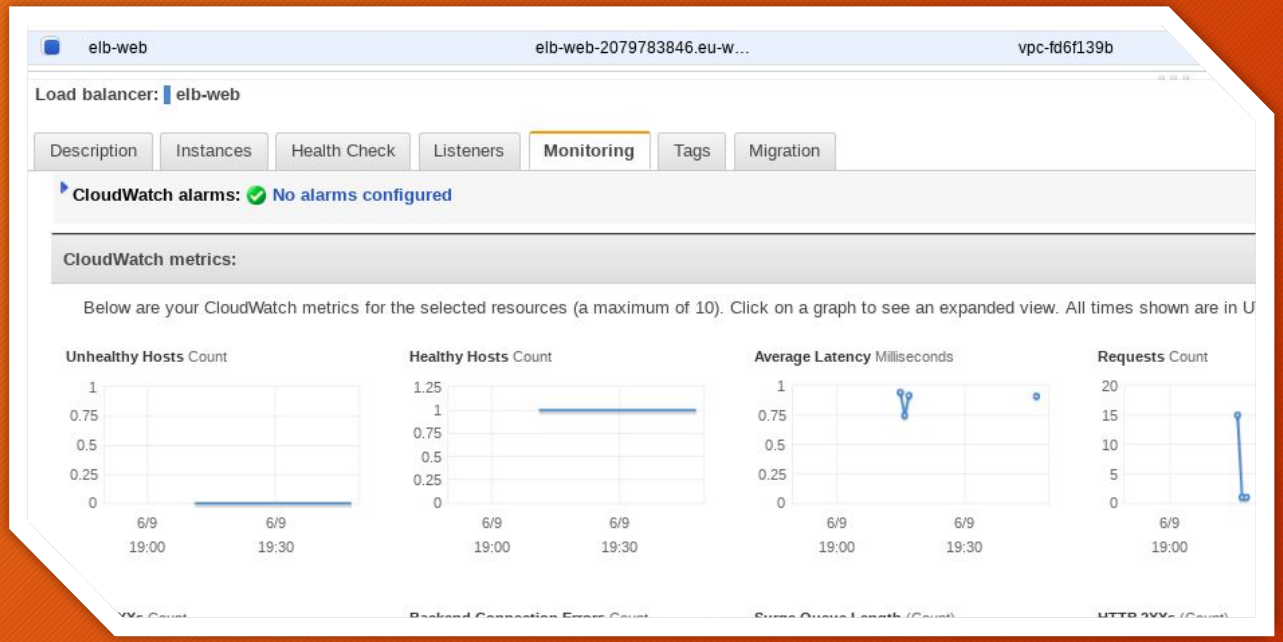
```
aws elb create-load-balancer \
     --load-balancer-name "elb-web" \
     --listeners "Protocol=HTTP,LoadBalancerPort=80,\
     InstanceProtocol=HTTP,InstancePort=80" \
     --subnets "subnet-88c87aee" "subnet-2364e76b" \
     --security-groups sg-15dd8e68
{
     "DNSName": "elb-web-2079783846.eu-west-1.elb…"
}
```

## 4. Asociar el Balanceador al ASG

```
aws autoscaling attach-load-balancers \
     --auto-scaling-group-name "web-servers" \
     --load-balancer-names "elb-web"
```

# CloudWatch

Es el servicio encargado de recopilar métricas y estadisticas del resto de servicios




¿Por qué no puedo monitorizar la memoria RAM o el espacio disponible en disco?

Scripts programados en **perl compatibles para** varios sistemas operativos en: mon-scripts

Video Demostrativo: /watch?v=1rfjP2PjWpU&t

# KOPS Y CLÚSTER AUTO-SCALER

Una herramienta para desplegar y administrar un clúster de Kubernetes sobre AWS manualmente

# ASPECTOS A TENER EN CUENTA

El número de nodos **MASTERS**

Diferentes zonas de disponibilidad

Elegir la red más óptima para nuestro caso

El tipo de instancia de los nodos

# CREACIÓN DEL CLÚSTER CON KOPS

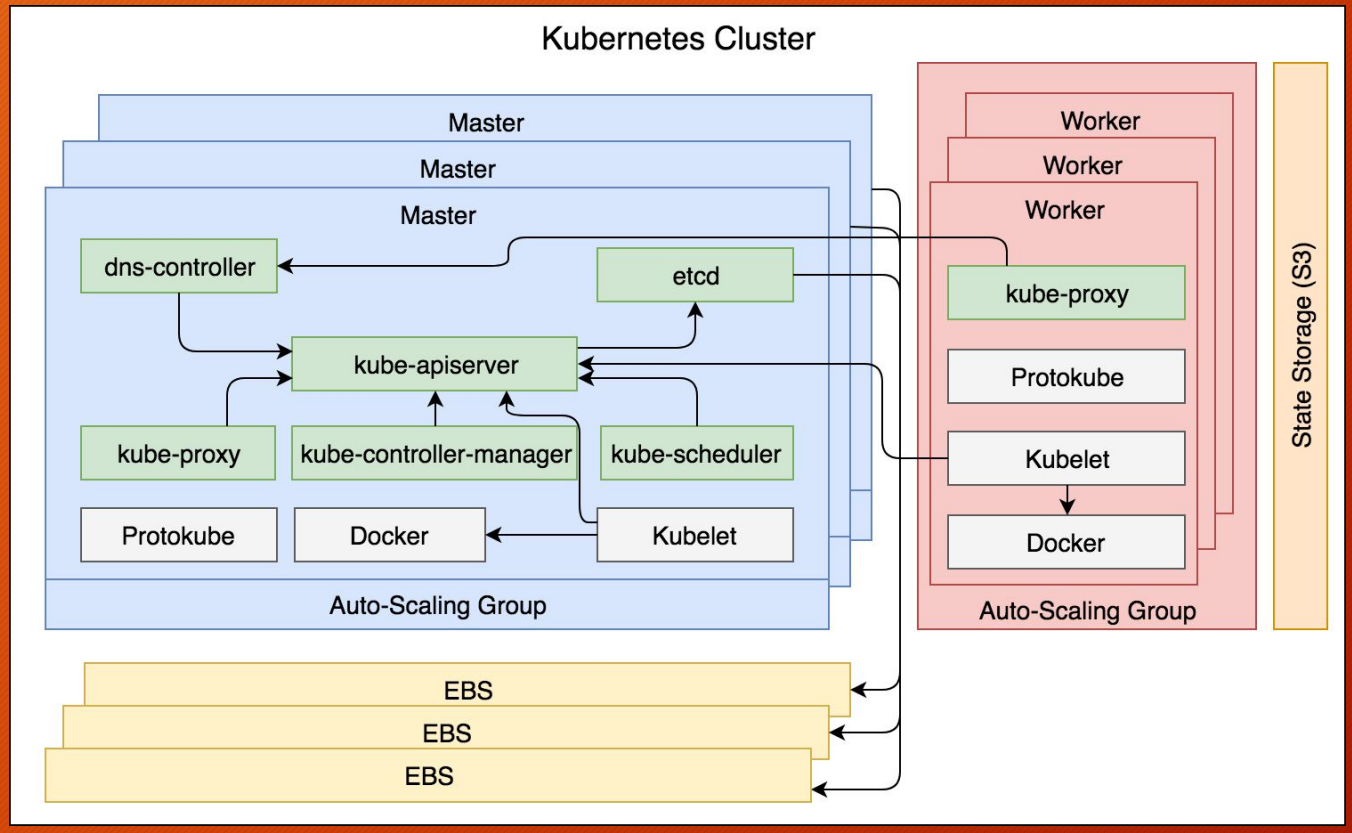
Creación del S3 Bucket

```
aws s3api create-bucket \
    --bucket “Proyecto-1234” \
    --create-bucket-configuration \
    LocationConstraint=eu-west-1
```

Creación del clúster

```
kops create cluster \
    --name “proyecto.k8s.local” \
    --master-count 3 \
    --node-count 1 \
    --node-size t2.micro \
    --master-size t2.small \
    --zones “eu-west-1a,eu-west-1b,..” \
    --master-zones “eu-west-1a,..” \
    --ssh-public-key proyecto.pub \
    --networking kubenet \
    --kubernetes-version v1.7.6 \
    --authorization RBAC \
    --yes
```

# Estructura del clúster en AWS



# ¿VIDEO?

# CONCLUSIÓN

Parece ser obvia, más demanda… más máquinas son necesarias.Y que mejor, que esto se encuentre programado.

Tanto AWS como Kubernetes individualmente, tiene muchas más opciones que no se han tratado en el proyecto.

¿Hacia donde ir?

Más conocimiento en AWS y la administración de Kubernetes, más conocimiento a nivel de definición de deployments, etc..

¿EKS?

# THANKS YOU!