

---

# **KUBERNETES, RANCHER Y AWS.**

Juan José Góngora Contreras

## Contents

<b>1</b>	<b>INTRODUCCION AL PROYECTO.</b>	<b>3</b>
1.1	CONTENEDORES Y ORQUESTADORES. . . . .	3
1.2	QUE ES KUBERNETES Y PORQUE USARLO. . . . .	3
1.3	CLUSTER DE KUBERNETES. . . . .	3
1.4	ARQUITECTURA DE KUBERNETES. . . . .	4
<b>2</b>	<b>MINIKUBE.</b>	<b>5</b>
2.1	OBJETOS DE KUBERNETES. . . . .	5
2.2	PARA LA GESTION DE KUBECTL . . . . .	5
2.3	TIPOS DE SERVICIOS . . . . .	5
2.4	NAMESPACES . . . . .	6
2.5	ESCALADO DE APLICACION. . . . .	6
<b>3</b>	<b>K3S</b>	<b>7</b>
3.1	INSTALACION DE SERVIDOR DE K3S . . . . .	7
3.2	CONFIGURACION DE KUBE EN UNA MAQUINA PARA MANEJAR EL CLUSTER. . . . .	8
<b>4</b>	<b>RANCHER</b>	<b>10</b>
4.1	INSTALACION DE RANCHER . . . . .	11
4.2	CONFIGURACION DE PERMISOS DE AWS. . . . .	11
4.3	CREACION DE UN USUARIO EN AWS Y PROPORCIONARLE PERMISOS. . . . .	11
4.4	AGREGAR CREDENCIALES DE AWS A RANCHER. . . . .	15
4.5	CONFIGURAR NODE TEMPLATES . . . . .	16
4.6	CREACION DEL CLUSTER . . . . .	17
4.7	LEVANTAR ALGUNA IMAGEN EN EL CLUSTER QUE HEMOS CREADO. . . . .	18
4.8	AUTO ESCALADO HORIZONTAL CON RANCHER. . . . .	18
<b>5</b>	<b>REFERENCIAS.</b>	<b>20</b>

## 1 INTRODUCCION AL PROYECTO.

Este proyecto consistirá en un aprendizaje del uso de contenedores y como orquestarlos entre máquinas. Para esto usaremos docker como gestor de contenedores y kubernetes como osquertador. El proyecto se dividirá en las siguientes partes.

- Introducción de gestor de contendor y orquestador.
- Entendimiento de como funciona Kubernetes y sus componentes mediante MiniKube.
- Creación de un clúster de Kubernetes en máquinas virtuales KVM.
- Creación de un clúster de alta disponibilidad con Rancher y AWS.

### 1.1 CONTENEDORES Y ORQUESTADORES.

Un contenedor es un proceso que ha sido aislado de todos los demás procesos en la máquina anfitriona. Ese aislamiento aprovecha características de Linux como los namespaces del kernel y cgroups. Aunque es posible tener más de un proceso en un contenedor las buenas prácticas nos recomiendan ejecutar sólo un proceso por contenedor (PID 1).

Algunas ventajas son que podemos tener bastantes contenedores en máquinas puesto que comparten el mismo núcleo el contenedor que la máquina anfitriona, levantar un contenedor es rápido y también nos proporcionan que son portátiles puesto que tenemos la imagen.

Entre los gestores de contenedores destacan: Docker, Podman y LXD.

Un orquestador es una herramienta o sistema que automatiza el despliegue, gestión, escalado, interconexión y disponibilidad de los contenedores.

Entre ellos tenemos: Docker Swarm, Kubernetes, Apache Mesos, OpenShift.

### 1.2 QUE ES KUBERNETES Y PORQUE USARLO.

Kubernetes es una plataforma de código abierto para el despliegue, escalado y gestión de aplicaciones contenedorizadas.

Es software libre, que aunque es complejo es de los mas destacados por los componentes que ofrece (Pods, services...)

Al ser tan destacada herramientas como Rancher la utilizan, y eso hace que no sea tan compleja porque Rancher se encarga de su gestión

Por lo principal se utiliza con docker que eso hace que se asegure un funcionamiento rápido y eficaz

### 1.3 CLUSTER DE KUBERNETES.

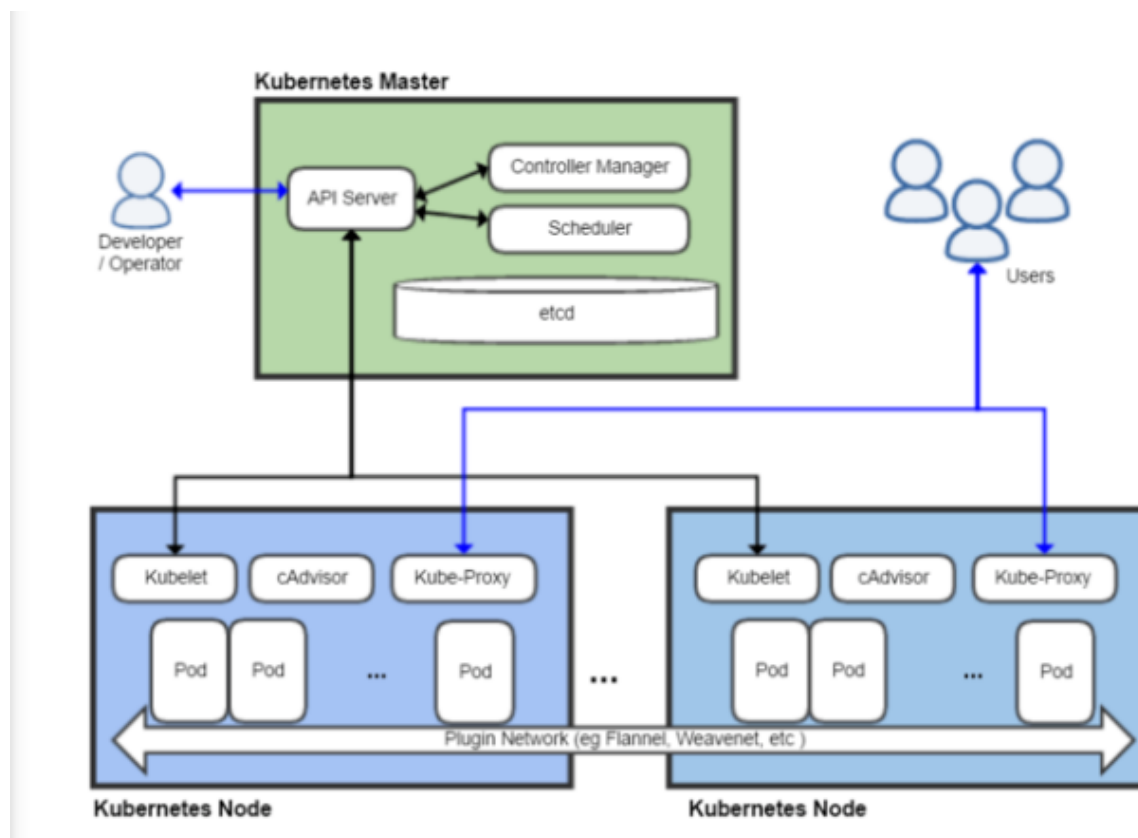
Esta compuesto por dos tipos de recursos.

- Master: Coordina todas las actividades del clúster como organizar aplicaciones, mantener el estado de aplicaciones, escalado, despliegue de actualizaciones. También recoge información de los nodos worker y los pods.
- Nodos: son workers que ejecutan las aplicaciones. Cada nodo contiene un agente llamado Kubelet que gestiona el nodo y mantiene la comunicación con el master.

En el despliegue de una aplicación en Kubernetes el master es el que inicia y organiza los contenedores para que se ejecuten en los nodos del clúster. La comunicación entre ellos se hace mediante la [API de Kubernetes](#).

## 1.4 ARQUITECTURA DE KUBERNETES.

En la siguiente imagen podemos ver los componentes mas importantes que tienen un master y un nodo.



- Plugins de red: Permiten la conexión entre pods de nodos diferentes y la integración de soluciones de red.
- Es una base de datos de clave-valor donde Kubernetes guarda todos los datos del clúster.
- API server: Componente del master que expone la API de Kubernetes
- Control manager: se encarga de comprobar si el estado deseado coincide con el de la realidad.
- Scheduler: Componente del master que observa que pods se han creado nuevos y no tienen nodo asignado, y les selecciona el nodo donde pueden ejecutarse.
- Kubelet: Agente que se ejecuta en cada nodo worker del clúster y que se asegura que los nodos están en ejecución y sanos. Kubelet no gestiona los pods que no han sido creados por kubernetes.

- Kube-proxy: Mantiene las reglas del networking en los nodos para los pods que se ejecutan en él de acuerdo con las especificaciones de los manifiestos.
- cAdvisor: Recoge los datos de uso de los contenedores.
- control plane: Nivel de orquestación de contenedores que expone la API para definir, desplegar y gestionar el ciclo de vida de los contenedores.
- Data plane: Nivel que proporciona los recursos, como CPU, memoria, red y almacenamiento, para que los pods se puedan ejecutar y conectar a la red.

## 2 MINIKUBE.

Es una implementación ligera de kubernetes que crea una máquina virtual localmente y despliega un clúster sencillo formado por un solo nodo. - minikube start - minikube dashboard

### 2.1 OBJETOS DE KUBERNETES.

Kubernetes tiene dos tipos de objetos, los básicos y los de nivel superior.

### 2.2 PARA LA GESTION DE KUBECTL

- Para lanzar un archivo YAML - `kubectl apply -f nginx.yaml`
- Para ver objetos: - `kubectl get all` - `kubectl get deployments` - `kubectl get nodes` - `kubectl get pods`
- Para eliminar objetos: - `kubectl delete deployment [nombre]` - `kubectl delete node [nombre]` - `kubectl delete pod [nombre]` - `kubectl delete -f [archivo.yaml]`

### 2.3 TIPOS DE SERVICIOS

- ClusterIP. - El servicio recibe una Ip interna a nivel de clúster y hace que el servicio solo sea accesible a nivel de clúster.
- NodePort. - Expone el servicio fuera del clúster concatenando la IP del nodo en el que esta el pod y un número de puerto entre 30000 y 32767, que es el mismo que todos los nodos.
- LoadBalancer. - Crea en cloud un balanceador externo con una IP externa asignada.
- ExternalName. - Expone el servicio usando un nombre.

Para crear uno de estos servicios hay que exponer el deployment y pasarle parámetro el tipo que queremos, por ejemplo yn tipo NodePort.

```
1 kubectl expose deployment jsonproducer --type=NodePort
```

## 2.4 NAMESPACES

Todos los objetos creados están en un mismo espacio, llamado default, si quisiéramos cambiar ese espacio para tenerlo todo mejor organizado, tendríamos que crearlos.

Creacion de un NameSpace.

```
1 kubectl create namespace juan
```

Creacion de un deployment asignando un NameSpace.

```
1 kubectl run nginxtote --image=nginx --port 80 --namespace juan
```

Cambiar namespace.

```
1 kubectl config set-context --current --namespace=juan
```

En este ultimo, todo lo que nos pongamos a hacer afecta solo a los objetos que estén en este NameSpace, todo lo que mostremos será a partir de esto, quiere decir que si en el NameSpace default tenemos un Deployment que llame hello-minikube, si cambiamos de NameSpace a **juan** por ejemplo, al mostrar nuestros deployments o pods no los mostrará.

## 2.5 ESCALADO DE APLICACION.

A la hora de crear un deployment se puede aumentar el número de replicas de pods que puede tener ese deployment, y kubernetes automáticamente irá creando tantos pods como pongamos en las diferentes máquinas que tenga acceso. Gracias a esto es posible la actualización de aplicaciones en caliente.

Si ejecutamos un deployment de prueba podemos ver lo siguiente.

```
1 kubectl apply -f nginx.yaml
```

Nos sale esto:

1	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
2	nginx	2/2	2	2	17s

- READY. - Nos dice el número de replicas(pods) tiene asignados y si están activos.
- UP-TO-DATE. - Nos indica número de replicas (pods) que están actualizados.
- AVAILABLE. - El número de replicas(pods) disponibles.

Ahora otro ejemplo.

Creamos un nuevo Deployment.

```
1 kubectl run jsonproducer --image=ualmtorres/jsonproducer:v0 --port 80
```

Y lo escalamos diciendo que queremos cuatro replicas.

```
1 kubectl scale deployments jsonproducer --replicas=4
```

Luego al obtener información ya sale que tenemos 4.

```
1 kubectl get deployments
```

1	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
2	jsonproducer	4/4	4	4	73m

Una vez hecho esto kubernetes ira enviando los equipos que quieran ver esta conexión a los diferentes workers, con un sistema de balanceo que tiene Kubernetes.

Si queremos bajar el número de pods nos bastaría con poner el mismo comando que usamos para escalar pero con menos números, se encargara de borrar los que vea oportunos.

## 3 K3S

### 3.1 INSTALACION DE SERVIDOR DE K3S

Para la instalación del servidor de K3S necesitamos tener como minio un master y un worker, yo lo haré con un Master y dos workers.

Cuando las máquinas estén listas tenemos que empezar desde la máquina `master` a realizar las operaciones, procedemos a descargar el paquete K3S, como K3S por así decirlo es un comando, lo que haremos sera descargarlo directamente en una ruta del PATH como por ejemplo `/usr/local/bin`

```
1 cd /usr/local/bin
```

Ahora descargamos el paquete.

```
1 wget https://github.com/rancher/k3s/releases/download/v0.2.0/k3s
```

Le proporcionamos permisos de ejecución.

```
1 chmod +x k3s
```

Para comenzar a desplegar el clúster con el nodo master tenemos que poner un comando muy sencillo.

```
1 k3s server &
```

Con esto el master ya esta listo y el clúster ya esta funcionando y podríamos ejecutar comandos `kubectl` siempre con el comando K3S por delante un ejemplo.

```
1 k3s kubectl get nodes
```

Y nos puede salir algo parecido a esto.

1	NAME	STATUS	ROLES	AGE	VERSION
2	master	Ready	<none>	27h	v1.13.4-k3s.1

Con esto el master esta listo, ahora tenemos que implementar los workers al clúster, empezamos accediendo a alguno de ellos y descargando el paquete y proporcionando permisos como en el `master` cuando todo eso este listo tenemos que ejecutar una instrucción.

```
1 k3s agent --server https://[IP_MASTER]:6443 --token [TOKEN_MASTER]
```

Nos encontramos con que no sabemos el token, esto lo podemos sacar en la máquina master en el siguiente archivo `/var/lib/rancher/k3s/server/node-token`

```
1 cat /var/lib/rancher/k3s/server/node-token
```

El contenido de ese archivo será el token que necesitamos entonces el comando de arriba en mi caso quedaría algo así.

```
1 k3s agent --server https://192.168.122.2:6443 --token
K10d6ccdb07ebdb6594f22aac004bfa36d6c0ed04d584bb295b8cdaf5c57ac582c5::node:5
a2356bfd0d35a9fd2dbc55041727a08
```

Tendremos que hacer lo mismo con el otro worker y de esta manera mediante el nodo master podríamos ejecutar las instrucciones `kubectl`

```
1 k3s kubectl get nodes
```

	NAME	STATUS	ROLES	AGE	VERSION
1	master	Ready	<none>	63m	v1.13.4-k3s.1
2	node1	Ready	<none>	36m	v1.13.4-k3s.1
3	node2	Ready	<none>	20m	v1.13.4-k3s.1

### 3.2 CONFIGURACION DE KUBE EN UNA MAQUINA PARA MANEJAR EL CLUSTER.

Para configurar El clúster creado por K3s se podría hacer desde el master pero hay una forma mas cómoda, que es poder configurarlo desde otra máquina cliente o local, para ello en esa máquina debemos tener instalado Kubectl y algunas cosas que nos pueden servir.

INSTALACION DE ALGUNOS PAQUETES QUE NOS HACEN FALTA.

```
1 apt update && apt install apt-transport-https curl git -y
```

INSTALACION DE KUBECTL

```
1 curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-key
add -
2 echo "deb https://apt.kubernetes.io/ kubernetes-xenial main" | sudo tee -a /etc
/apt/sources.list.d/kubernetes.list
3 apt-get update
4 apt install kubectl -y
```

Cuando este instalado lo configuraremos para que coja todas las credenciales y demás del nodo master, para ello lo hacemos mediante un archivo de configuración, creamos en nuestra carpeta personal una carpeta que se llame `.kube` de ahí nos conectamos al master mediante scp para coger un archivo de configuración que se encuentra en `/etc/rancher/k3s` y se llama `k3s.yaml`. Lo cogemos a nuestra máquina con el nombre de config, de la siguiente manera.

```
1 mkdir .kube
```



```
2 cd .kube
3 scp [IP_MASTER]:/etc/rancher/k3s/k3s.yaml config
```

Cuando lo tengamos tendremos que editarlo porque esta configurado para la máquina local, tendremos que decirle donde se encuentra el master, así que la línea siguiente

```
1 https://localhost:6443
```

Cambiamos localhost por la ip del master en mi caso

```
1 server: https://192.168.122.2:6443
```

Cuando lo tengamos configurado solo nos falta general una variable de entorno para kubernetes y decirle que todos los comandos realizados coja información del archivo que acabamos de configurar, de la siguiente forma.

```
1 export KUBECONFIG=~/.kube/config
```

De este modo si ahora probamos a ejecutar la orden `kubectl get nodes` nos saldrá la información de los nodos como podría ser esta

1	NAME	STATUS	ROLES	AGE	VERSION
2	master	Ready	<none>	63m	v1.13.4-k3s.1
3	node1	Ready	<none>	36m	v1.13.4-k3s.1
4	node2	Ready	<none>	20m	v1.13.4-k3s.1

Ahora podríamos hacer cualquier cosa con kubernetes y el master estaría balanceado entre los nodos

Como prueba podemos crear por ejemplo un deployment con nginx

```
1 kubectl create deploy nginx --image=nginx
```

Si obtenemos la información de los deployments y los pods con la siguiente orden

```
1 kubectl get deployments,pod
```

Podemos ver algo parecido a esto

1	NAME	READY	UP-TO-DATE	AVAILABLE	AGE
2	deployment.extensions/nginx	1/1	1	1	85s
3					
4	NAME	READY	STATUS	RESTARTS	AGE
5	pod/nginx-5c7588df-kklmn	1/1	Running	0	85s

Para la conexión a ese pod que hemos creado vamos por medio de la IP del nodo master y creando un servicio de tipo nodeport

```
1 kubectl expose deploy nginx --port=80 --type=NodePort
```

Cuando esto termine podemos ver en que puerto se ha mapeado mostrando los servicios con kubectl

```
1 kubectl get services
```

Ahora podríamos probar a escalar la aplicación y que ejecuten varios pods.

```
1 kubectl scale --replicas=3 deploy/nginx
```

Y si mostramos los pods

1	NAME	READY	STATUS	RESTARTS	AGE
2	nginx-5c7588df-54ls6	1/1	Running	0	37s
3	nginx-5c7588df-kklmn	1/1	Running	0	12m
4	nginx-5c7588df-vf8gt	1/1	Running	0	37s

También podríamos ver en que nodo se esta ejecutando cada uno.

```
1 kubectl get pod -o wide
```

1	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
	NOMINATED NODE	READINESS GATES					
2	nginx-5c7588df-54ls6	1/1	Running	0	2m35s	10.42.1.3	node1
	<none>	<none>					
3	nginx-5c7588df-kklmn	1/1	Running	0	14m	10.42.2.2	node2
	<none>	<none>					
4	nginx-5c7588df-vf8gt	1/1	Running	0	2m35s	10.42.0.6	master
	<none>	<none>					

Si hacemos un cuarto escalado podríamos ver que una de los nodos á dos, en mi caso el nodo1

1	NAME	READY	STATUS	RESTARTS	AGE	IP	NODE
	NOMINATED NODE	READINESS GATES					
2	nginx-5c7588df-4wl5t	1/1	Running	0	9s	10.42.1.4	node1
	<none>	<none>					
3	nginx-5c7588df-54ls6	1/1	Running	0	4m56s	10.42.1.3	node1
	<none>	<none>					
4	nginx-5c7588df-kklmn	1/1	Running	0	17m	10.42.2.2	node2
	<none>	<none>					
5	nginx-5c7588df-vf8gt	1/1	Running	0	4m56s	10.42.0.6	master
	<none>	<none>					

## 4 RANCHER

Rancher es una aplicación Web que nos permite el uso de Kubernetes de una forma gráfica, usándolo en un servidor privado con nuestra infraestructura o clouds de Internet como pueden ser: - Amazon Web Services. - Azure. - Google container engine. - DigitalOcean - Custom

Rancher es una pila completa de software para equipos que adoptan contenedores. Aborda los desafíos operativos y de seguridad de administrar múltiples clústeres de Kubernetes en cualquier infraestructura, al tiempo que proporciona a los equipos de DevOps herramientas integradas para ejecutar cargas de trabajo en contenedores.

Algunas de las características de Rancher podrían ser:

- Gestión centralizada de cualquier grupo de Kubernetes
- Una plataforma empresarial para administrar Kubernetes en todas partes

- Sin ninguna plataforma de nube, puede ser personalizado
- Aprovisionamiento de clúster

## 4.1 INSTALACION DE RANCHER

Para esto lanzaremos una máquina ec2 en nuestro AWS

NOTA: PARA QUE RANCHER FUNCIONE MINIMO TIENE QUE SER UNA MAQUINA DE t2.small

Cuando esa máquina este disponible tendremos que actualizar la máquina e instalar docker.

```
1 apt update && apt install docker.io -y
```

Cuando lo tengamos bastara con lanzar un contendor en docker y podremos acceder a rancher desde el navegador.

```
1 docker run -d -p 80:80 -p 443:443 rancher/rancher
```

Nos pedirá que necesitamos proporcionar una clave para acceder al administrador de rancher

NOTA:RECOMIENDO QUE AUNQUE SEA UNA PRUEBA LA CLAVE SEA SEGURA PORQUE MAS TARDE LE DAREMOS A RANCHER NUESTRAS CEDENCIALES Y CUALQUIERA PODRIA HACER COSAS SI ACCEDEN CON UNA CLAVE FACIL.

Después de eso nos dejara acceder a rancher perfectamente.

## 4.2 CONFIGURACION DE PERMISOS DE AWS.

Para que todo lo que queremos hacer funcione necesitamos gestionar un servicio que nos ofrece AWS que se trata de IAM, con esto crearemos un usuario para rancher que le permita el acceso a nuestro AWS.

## 4.3 CREACION DE UN USUARIO EN AWS Y PROPORCIONARLE PERMISOS.

Lo primero es crear el usuario que eso debemos de ir a los servicios de AWS y buscar IAM, allí a la seccion de usuario y como no tenemos ningún usuario nos sugiere crear uno.

Rellenamos el nombre de usuario y después tenemos que seleccionar que tipo de acceso tendrá ese usuario, seleccionaremos el segundo, que s mediante consola. La clave podemos escoger una o que la genere automática, sugiero automática.

Añadir usuario(s)

Establecer los detalles del usuario

Puede añadir varios usuarios a la vez con los mismos permisos y el mismo tipo de acceso. [Más información](#)

Nombre de usuario\*

[Añadir otro usuario](#)

Seleccionar el tipo de acceso de AWS

Seleccione la forma en que estos usuarios accederán a AWS. Las claves de acceso y las contraseñas generadas automáticamente se proporcionan en el último paso. [Más información](#)

Tipo de acceso\*

- ☐ Acceso mediante programación  
Habilita una ID de clave de acceso y una clave de acceso secreta para el SDK, la CLI y la API de AWS, además de otras herramientas de desarrollo.
- ☒ Acceso a la consola de administración de AWS  
Habilita una contraseña que permite a los usuarios iniciar sesión en la consola de administración de AWS.

Contraseña de la consola\*

- ☒ Contraseña generada automáticamente
- ☐ Contraseña personalizada

Requerir el restablecimiento de contraseña ☒ El usuario debe crear una contraseña nueva en el próximo inicio de sesión. Los usuarios obtienen automáticamente la política `IAMUserChangePassword` que les permite cambiar su propia contraseña.

\* Obligatorio

[Cancelar](#) [Siguiente: Permisos](#)

Seguidamente nos pedirá que lo metamos en un grupo para darle permisos, esto por ahora lo dejamos, lo haremos mas tarde. Seguimos creando el usuario dándole a siguiente hasta terminar todos los paso no hay que rellenar nada mas.

Cuando hagamos esto pasaremos a crear políticas y roles de IAM ([Permisos](#))

Crearemos una política para el [controlpane](#) de nuestro futuro clúster de kubernetes para crear una política iremos a IAM, política, crear política, hay dos formas de hacer esto que es ir seleccionando que permisos queremos proporcionar y el otro mediante un archivo JSON que es lo mismo que lo otro pero es en código os mostrare como seria el de código.

```

1  {
2    "Version": "2012-10-17",
3    "Statement": [
4      {
5        "Effect": "Allow",
6        "Action": [
7          "autoscaling:DescribeAutoScalingGroups",
8          "autoscaling:DescribeLaunchConfigurations",
9          "autoscaling:DescribeTags",
10         "ec2:DescribeInstances",
11         "ec2:DescribeRegions",
12         "ec2:DescribeRouteTables",
13         "ec2:DescribeSecurityGroups",
14         "ec2:DescribeSubnets",
15         "ec2:DescribeVolumes",
16         "ec2:CreateSecurityGroup",
17         "ec2:CreateTags",
18         "ec2:CreateVolume",
19         "ec2:ModifyInstanceAttribute",
20         "ec2:ModifyVolume",
21         "ec2:AttachVolume",
22         "ec2:AuthorizeSecurityGroupIngress",
23         "ec2:CreateRoute",

```

```

24         "ec2:DeleteRoute",
25         "ec2:DeleteSecurityGroup",
26         "ec2:DeleteVolume",
27         "ec2:DetachVolume",
28         "ec2:RevokeSecurityGroupIngress",
29         "ec2:DescribeVpcs",
30         "elasticloadbalancing:AddTags",
31         "elasticloadbalancing:AttachLoadBalancerToSubnets",
32         "elasticloadbalancing:ApplySecurityGroupsToLoadBalancer",
33         "elasticloadbalancing:CreateLoadBalancer",
34         "elasticloadbalancing:CreateLoadBalancerPolicy",
35         "elasticloadbalancing:CreateLoadBalancerListeners",
36         "elasticloadbalancing:ConfigureHealthCheck",
37         "elasticloadbalancing>DeleteLoadBalancer",
38         "elasticloadbalancing>DeleteLoadBalancerListeners",
39         "elasticloadbalancing:DescribeLoadBalancers",
40         "elasticloadbalancing:DescribeLoadBalancerAttributes",
41         "elasticloadbalancing:DetachLoadBalancerFromSubnets",
42         "elasticloadbalancing:DeregisterInstancesFromLoadBalancer",
43         "elasticloadbalancing:ModifyLoadBalancerAttributes",
44         "elasticloadbalancing:RegisterInstancesWithLoadBalancer",
45         "elasticloadbalancing:SetLoadBalancerPoliciesForBackendServer",
46         "elasticloadbalancing:AddTags",
47         "elasticloadbalancing:CreateListener",
48         "elasticloadbalancing:CreateTargetGroup",
49         "elasticloadbalancing>DeleteListener",
50         "elasticloadbalancing>DeleteTargetGroup",
51         "elasticloadbalancing:DescribeListeners",
52         "elasticloadbalancing:DescribeLoadBalancerPolicies",
53         "elasticloadbalancing:DescribeTargetGroups",
54         "elasticloadbalancing:DescribeTargetHealth",
55         "elasticloadbalancing:ModifyListener",
56         "elasticloadbalancing:ModifyTargetGroup",
57         "elasticloadbalancing:RegisterTargets",
58         "elasticloadbalancing:SetLoadBalancerPoliciesOfListener",
59         "iam:CreateServiceLinkedRole",
60         "kms:DescribeKey"
61     ],
62     "Resource": [
63         "*"
64     ]
65 }
66 ]
67 }

```

Lo que estamos haciendo en esta política es darle a todo aquel que este asociado a esta política el poder de los permisos como ejecutar instancias, borrarlas, crear security groups y todo lo relacionado para la gestión que necesita rancher para la creación del clúster.

Después revisamos la política le ponemos un nombre por ejemplo `controlpane_policy` y la creamos.

Ahora tendremos que hacer una para el `etcd` y `worker`, el mismo proceso pero el contenido del JSON es diferente, es este

```

1 {
2     "Version": "2012-10-17",

```

```

3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "ec2:DescribeInstances",
8                 "ec2:DescribeRegions",
9                 "ecr:GetAuthorizationToken",
10                "ecr:BatchCheckLayerAvailability",
11                "ecr:GetDownloadUrlForLayer",
12                "ecr:GetRepositoryPolicy",
13                "ecr:DescribeRepositories",
14                "ecr:ListImages",
15                "ecr:BatchGetImage"
16            ],
17            "Resource": "*"
18        }
19    ]
20 }

```

La revisamos y por ejemplo la podemos llamar `etcd_worker_policy`

Lo siguiente es crear roles y asociarlos cada uno a su política. Para ellos nos vamos a IAM, roles y nuevo rol. Aquí asociaremos la política para cada uno de los roles, por ejemplo si estamos creando el rol de `controlpane` le asociamos la política de `controlpane_policy` y a ese rol podemos llamarle `controlpane_role`. Creamos también el rol para `etcd_worker_policy` y la llamamos `etcd_worker_role`.

Importante crear otra política que una estas dos, es decir, creamos otro rol que y le asociamos `controlpane_policy` y `etcd_worker_policy` y por ejemplo podemos llamarla `all_role`

Ahora volveríamos a la creacion de política para crear la ultima política que nos hace falta `PASSROLE`

Necesitaremos lo siguiente - ID de nuestra cuenta de AWS, esto lo podemos encontrar dando en nuestro nombre de usuario arriba a la derecha, mi cuenta. - REGION esto lo podemos encontrar arriba a la derecha donde pone al lado de nuestro nombre de usuario encontraremos el estado donde estamos y si pulsamos nos dirá nuestra región, en mi caso es `us-east-2`

En el siguiente archivo JSON hay datos que hay que cambiar dependiendo de nuestra `REGION`, `ID_CUENTA`, `NOMBRE DE ROLES`

```

1  {
2      "Version": "2012-10-17",
3      "Statement": [
4          {
5              "Sid": "VisualEditor0",
6              "Effect": "Allow",
7              "Action": [
8                  "ec2:AuthorizeSecurityGroupIngress",
9                  "ec2:Describe*",
10                 "ec2:ImportKeyPair",
11                 "ec2:CreateKeyPair",
12                 "ec2:CreateSecurityGroup",
13                 "ec2:CreateTags",
14                 "ec2>DeleteKeyPair"
15             ],
16             "Resource": "*"

```

```

17     },
18     {
19         "Sid": "VisualEditor1",
20         "Effect": "Allow",
21         "Action": [
22             "ec2:RunInstances",
23             "iam:PassRole"
24         ],
25         "Resource": [
26             "arn:aws:ec2:REGION::image/ami-*",
27             "arn:aws:ec2:REGION:ID_CUENTA:instance/*",
28             "arn:aws:ec2:REGION:ID_CUENTA:placement-group/*",
29             "arn:aws:ec2:REGION:ID_CUENTA:volume/*",
30             "arn:aws:ec2:REGION:ID_CUENTA:subnet/*",
31             "arn:aws:ec2:REGION:ID_CUENTA:key-pair/*",
32             "arn:aws:ec2:REGION:ID_CUENTA:network-interface/*",
33             "arn:aws:ec2:REGION:ID_CUENTA:security-group/*",
34             "arn:aws:iam:ID_CUENTA:role/etcd_worker_role",
35             "arn:aws:iam:ID_CUENTA:role/controlpane_role"
36         ]
37     },
38     {
39         "Sid": "VisualEditor2",
40         "Effect": "Allow",
41         "Action": [
42             "ec2:RebootInstances",
43             "ec2:TerminateInstances",
44             "ec2:StartInstances",
45             "ec2:StopInstances"
46         ],
47         "Resource": "arn:aws:ec2:REGION:ID_CUENTA:instance/*"
48     }
49 ]
50 }

```

Cuando tengamos esto solo nos faltaría volver a los roles y crear el rol para la política de `PASSROLE` como hemos hecho con las políticas anteriores.

Lo siguiente como nos habíamos dejado el usuario sin darle los permisos ahora es el momento ya que los acabamos de hacer, nos tendríamos que ir a IAM, usuarios, “nuestro usuario”, agregar permisos, crear grupo nos saldrá la lista de políticas, aquí agregamos las tres políticas que hemos creado antes `controlpane_policy`, `etcd_worker_policy` y `passrole_policy`. Por ejemplo le podemos llamar `policy_group`

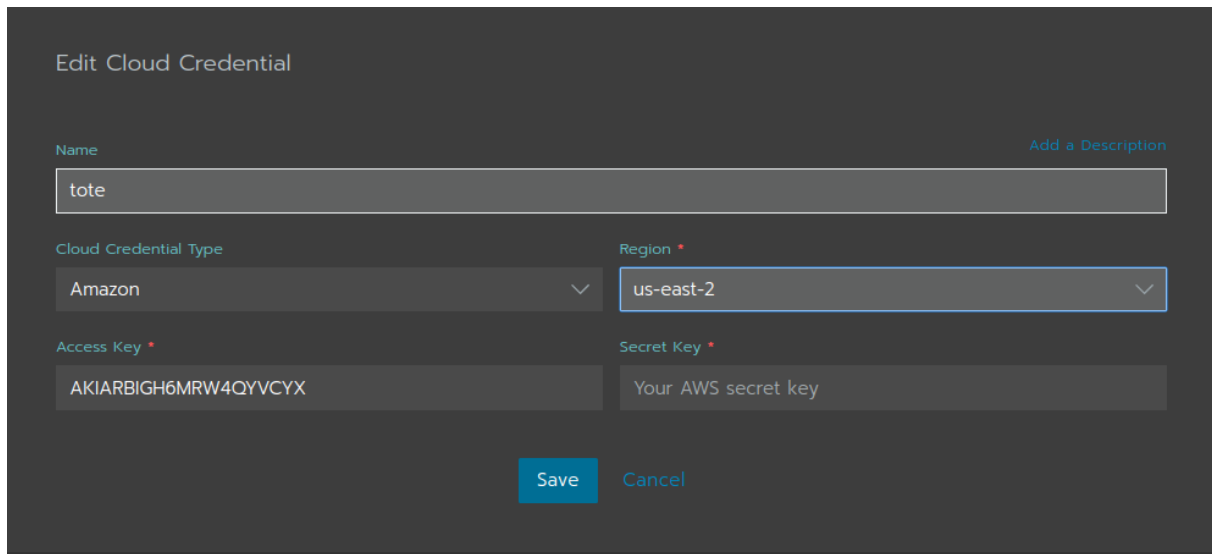
#### 4.4 AGREGAR CREDENCIALES DE AWS A RANCHER.

Cuando nos vayamos a rancher tenemos que hacer dos cosas previas darle credenciales de AWS del usuario que hemos creado, en rancher nos vamos a `cloud credentials`, `add cloud credential`.

Nos pedirá el nombre, tipo de nube(Amazon), nuestra región(Como la que hemos puesto en las políticas), access key y secret key.

Para con seguir el `access key` y `secret key` debemos de ir a AWS y entrar en IAM, usuarios, credenciales de seguridad y le debemos dar a crear una nueva clave de acceso entonces AWS nos dará ambas cosas y las

ponemos en rancher.



## 4.5 CONFIGURAR NODE TEMPLATES

Lo que vamos a configurar a continuación es las características de máquinas de los nodos, al darle a agregar nodo nos dirá que tipo de Nube que es Amazon, la región y nuestra cloud credentials las zonas de red que hay en nuestra región ponemos las que queramos y la subred cogemos la que tenemos por defecto. Cuando le demos a siguiente tocarán los puertos abiertos para la máquina que ponemos que default de rancher esto abrirá los que rancher crea oportunos.

Lo siguiente son las especificaciones de la máquina que podemos escoger la instancia que queramos. - AMI: Aquí tendremos que poner una ami de rancher que la ami cambia según al región, dejo un enlace para que se vea la ami según tu región

```
1 https://github.com/rancher/os/blob/master/README.md/#user-content-amazon
```

- IAM INSTANCE PROFILE NAME: Aquí ponemos el nombre del rol en el que hemos juntado las dos políticas antes, el que yo he llamado `all_role`
- SSH User: Aquí tenemos que poner `rancher`



**4. Instance**  
Customize the EC2 Instance that will be created.

Instance Type: **t2.medium**

Spot Instance: ☐ Request spot instance

Root Disk Size: **16** GB

Encryption: ☐ Encrypt EBS Volume

AMI: **ami-002ab867b8b8591d5**

IAM Instance Profile Name: **all\_role**

SSH User: **rancher**

Private IP: ☐ Use only private IP address

**+ Add AWS Tag**

RANCHER TEMPLATE

Name: **ALL**

Add a Description

Y por ultimo poner el nombre del node.

Para que sirva esto de NODE template realmente. por ejemplo si queremos que el master o los worker sean máquinas diferentes configuramos la máquina de una forma u otra creando varios.

## 4.6 CREACION DEL CLUSTER

Ya por ultimo crear el clúster, nos vamos a clúster y add clúster, seleccionamos el ec2 y le ponemos nombre a nuestro clúster

Ponemos un prefijo a la máquina, seleccionamos el template que hayamos creado y luego tenemos que seleccionar que nodos serán `etcd`, `controlplane` o `worker` podemos poner hasta cuantas queremos crear.

Cluster Name: **Cluster**

Add a Description

Name Prefix	Count	Template	Auto Replace	etcd	Control Plane	Worker	Taints
Master	1	ALL	0 minutes	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input type="checkbox"/>	Taints
Worker	1	ALL	0 minutes	<input type="checkbox"/>	<input type="checkbox"/>	<input checked="" type="checkbox"/>	Taints

Number of nodes required: 1, 3, or 5

**+ Add Node Pool**

Luego mas abajo nos pondrá que cloud provider, deberemos seleccionar que Amazon y damos en create y el clúster se empieza a crear.

## 4.7 LEVANTAR ALGUNA IMAGEN EN EL CLUSTER QUE HEMOS CREADO.

Para hacer un deploy tenemos que irnos a nuestro clúster, system. Desde ahí a la derecha nos sale deploy y seguidamente la configuración del deploy como su nombre, su docker image, si queremos crearle algún servicio y tal.

Mas abajo tendremos las variables que tendrá la imagen de docker, los volúmenes que podemos usar con `secret` o `config map`. Como un archivo yaml pero en modo gráfico, si queremos importat nuestro archivo porque ya lo tenemos listo también lo podemos hacer. Y para finalizar simplemente le damos a launch.

Si por ejemplo hemos creado un nodeport y queremos acceder a esa máquina desde el exterior en el menú de System nos sale nuestro nuevo pod que si nos fijamos debajo de su nombre tenemos el puerto que ha generado rancher random o si hemos puesto nosotros uno manual, solo tenemos que o bien darle ahí y nos lleva automáticamente o también si queremos acceder nosotros con ip tendríamos que coger la IP del MASTER junto a ese puerto.

State	Name	Image	Scale
Namespace: cattle-system			
Active	apache	httpd 1 Pod / Created a minute ago / Pod Restarts: 0	1
Active	cattle-cluster-agent	rancher/rancher-agent:v2.4.3 1 Pod / Created 37 minutes ago / Pod Restarts: 0	1
Active	cattle-node-agent	rancher/rancher-agent:v2.4.3 3 Pods / Created 37 minutes ago / Pod Restarts: 0	1 per node
Active	kube-api-auth	rancher/kube-api-auth:v0.1.4 1 Pod / Created 37 minutes ago / Pod Restarts: 0	1 per node

## 4.8 AUTO ESCALADO HORIZONTAL CON RANCHER.

Esto se hace mediante HPA, en los pods hay que limitarles los milicpu, sino no nos dejara ponerle un HPA.

Al crear un Deploy tenemos que ver las opciones avanzadas de la creacion e irnos a `Security & Host config` y aquí bajar hasta encontrar CPU reservation y limitarlas CPUs. Si lo dejamos sin reservar HPA no podrá realizar el escalado, para probar podemos reservar pocas, por ejemplo 100 o 200 y darle a Launch.

Memory Reservation: e.g. 128 MiB

Memory Limit: ☒ No Limit ☐ Limit to 128 MiB

CPU Reservation: 200 milli CPUs

CPU Limit: ☒ No Limit ☐ Limit to 1000 milli CPUs

NVIDIA GPU Reservation: e.g. 1 GPUs

Seguidamente tenemos que ir a System, resources, HPA, add HPA.

Le pondremos un nombre, seleccionamos en que namespace se encuentra, y nuestro workload(pod), nuestras replicas como mínimo y máximo.

Y ya abajo la métrica, que puede ser o de CPU o re memoria RAM, recomiendo la de porcentaje de CPU

Add Horizontal Pod Autoscaler

Name: apache\_hpa Add a Description Namespace: cattle-system

Workload: apache

Min Replicas: 1 Max Replicas: 10

Metrics

Metric type: Resource Metric Name: CPU

Target Type: Average Utilization Quantity: 40%

+ Add Metric

Show advanced options

Create Cancel

Ahora, como podemos probar esto, muy fácil, con la herramienta de apache `apache benchmark` que sino lo tienes instalado en tu Linux basta con lo siguiente.

```
1 apt install apache2-utils
```

Este comando funciona de la siguiente manera, envía un total de peticiones, y también las que le digamos simultáneamente, se vería de la siguiente forma.

```
1 ab -n100000 -c100 http://IP_SERVER/
```

En este caso envía 100000 peticiones y va enviando 100 simultáneamente, a la dirección que tengamos nuestro pod (Dirección NodePort)

Cuando hagamos esto el HPA si ve que sobrepasa el limite establecido lo ira escalando cuanto sea necesario hasta el máximo que pongamos.

HPA

Download YAML Delete

Search

State Name Target Workload Current Replicas Last Scale

Namespace: cattle-system

Active apache\_hpa apache 1 293m / 146% / 40% Desired 4 / Replicas Range 1 ~ 10 a few seconds ago

```
root@ip-172-31-17-113: /home/ubuntu
Archivo Editar Ver Buscar Terminal Ayuda
HTML transferred: 4500000 bytes
Requests per second: 4263.94 [#/sec] (mean)
Time per request: 70.357 [ms] (mean)
Time per request: 0.235 [ms] (mean, across all concurrent requests)
Transfer rate: 1203.40 [Kbytes/sec] received

Connection Times (ms)
      min      mean[+/-sd] median   max
Connect:    0      28 150.9      4    3050
Processing:  1      42  47.7     28    922
Waiting:    0      28  36.5     23    900
Total:      1      70 160.1     34   3090

Percentage of the requests served within a certain time (ms)
 50%    34
 66%    42
 75%    50
 80%    58
 90%   102
 95%   103
 98%   1031
 99%   1060
100%   3090 (longest request)

root@ip-172-31-17-113: /home/ubuntu#
```

En la siguiente imagen lo he sobrepasado y según el porcentaje, ha tenido que crear otros 4 pods idénticos.

## 5 REFERENCIAS.

- [Documentación de kubernetes](#)
- [Documentación de AWS](#)
- [Documentación de Rancher](#)
- [Kubernetes. Un orquestador de contenedores que debes poner en tu vida](#)
- [How to Build a Kubernetes Clúster With EC2 and Rancher 2](#)