LABORATORIO DE ADMINITRACIÓN Y GESTIÓN DE REDES Y SISTEMAS

Manuel Carmona Acebedo
Grado en Ingeniería Telemática
Universidad Rey Juan Carlos

Índice

1. INTRODUCCIÓN

El propósito de este trabajo será el de desplegar y orquestar un servicio web sobre un entorno cloud.

Para ello usaremos una aplicación de prueba, escrita en Go, que ofrece una interfaz REST para guardar “usuarios ” en una base de datos.

La base de datos estará ubicada en AWS (Amazon Web Services) a través de la utilidad Amazon Relational Database Service (RDS).

Containizaremos la aplicación con Docker, para después desplegarla en un cluster local configurado con la herramienta minikube.

La orquestación se hará con kubernetes pudiendo desplegar, actualizar, escalar o eliminar nuestra aplicación.

Para probar nuestro servicio utilizaremos la extensión de Chrome, Postman, que nos permite formar peticiones http, mientras que para conectar con la base de datos usaremos el cliente psql.

Todo el código y archivos de configuración necesarios para este trabajo se encuentren en https://github.com/mcarmonaa/lagrs

Además también puedes encontrar el vídeo donde se explica y se realiza todo este proceso en el repositorio. En él se explica todo con detalle, este documento es para poner en contexto el trabajo e introducir las herramientas usadas.

2. SOFTWARE Y HERRAMIENTAS

El OS usado es:

```
[manuel@manubook ~]$ uname -a
Linux manubook 4.4.0-59-generic #80~14.04.1-Ubuntu SMP Fri Jan 6 18:02:02 UTC
2017 x86_64 x86_64 x86_64 GNU/Linux
```

A continuación se presenta una lista del software y herramientas necesarias y un enlace a la documentación neceraria donde explica como instalarlo. Entre paréntesis se indica la versión de la herramienta usada en este trabajo:

- Go (1.7.4): https://golang.org/doc/install
- Postman (4.9.2): https://chrome.google.com/webstore/detail/postman/fhbjgbiflinjbdggehcddcbncdddomop?hl=es
- AWS account: https://aws.amazon.com/
- psql (9.6.1): https://help.ubuntu.com/community/PostgreSQL
- Docker (1.12.6): https://docs.docker.com/engine/installation/linux/ubuntulinux/
- Virtual Box(4.3.36_Ubuntur105129): https://www.virtualbox.org/wiki/Linux_Downloads
- kubectl (v1.4.4): https://kubernetes.io/docs/getting-started-guides/kubectl/
- minikube (v0.14.0): https://github.com/kubernetes/minikube/releases

3. APLICACIÓN

El servicio web que se va a desplegar, es una aplicación muy sencilla programada en Go, que expone una interfaz REST de la siguiente manera:

/users , GET (devuelve la lista de usuarios)
/users, POST (Crea un usuario)
/users/{id}, GET (devuelve un usario)
/users/{id}, PUT (actualiza un usuario)
/users/{id}, DELETE (elimina un usuario)

El intercambio de información se realiza en formato JSON, por lo que las peticiones deberán llevar la cabecera “Content-Type: application/json”. Esto lo haremos con el clienet REST Postman.

Un objeto usuario en formato JSON, tiene este aspecto:

```
{
	"name": "Manuel",
	"age": 26,
	"mail": "manu@mail.com",
	"role": 1
}
```

La aplicación está dividida en dos microservicios que se comunicarán a través del framework de RPCs llamado gRPC (http://www.grpc.io/) desarrollado por Google. La serialización de la información se hará con el mecanismo, también de Google, Protocol Buffers (https://developers.google.com/protocol-buffers/).

El primer servicio será el punto de entrada que expondrá la interfaz (https://github.com/mcarmonaa/lagrs/tree/master/gserver), mientras que el segundo servicio (https://github.com/mcarmonaa/lagrs/tree/master/serviceusers) servirá de backend, manejando todas las operaciones sobre usuarios y conectando contra la base de datos.

La base de datos será una Postgresql levantada en aws-rds, y para conectarnos desde la aplicación usaremos un ORM para Go que se llama Gorm (https://github.com/jinzhu/gorm)

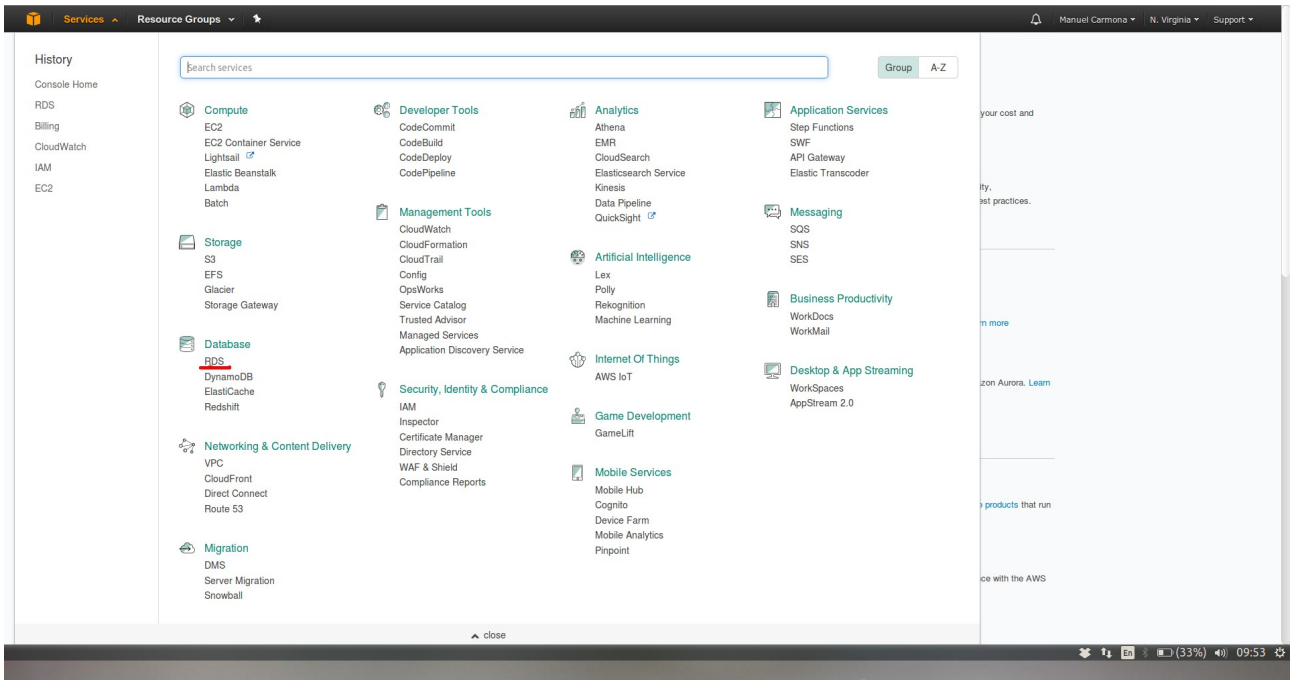
Por otro lado, para conectarnos a la base de datos desde fuera de la aplicación y poder ver las tablas, usaremos el cliente psql.

4. AWS-RDS

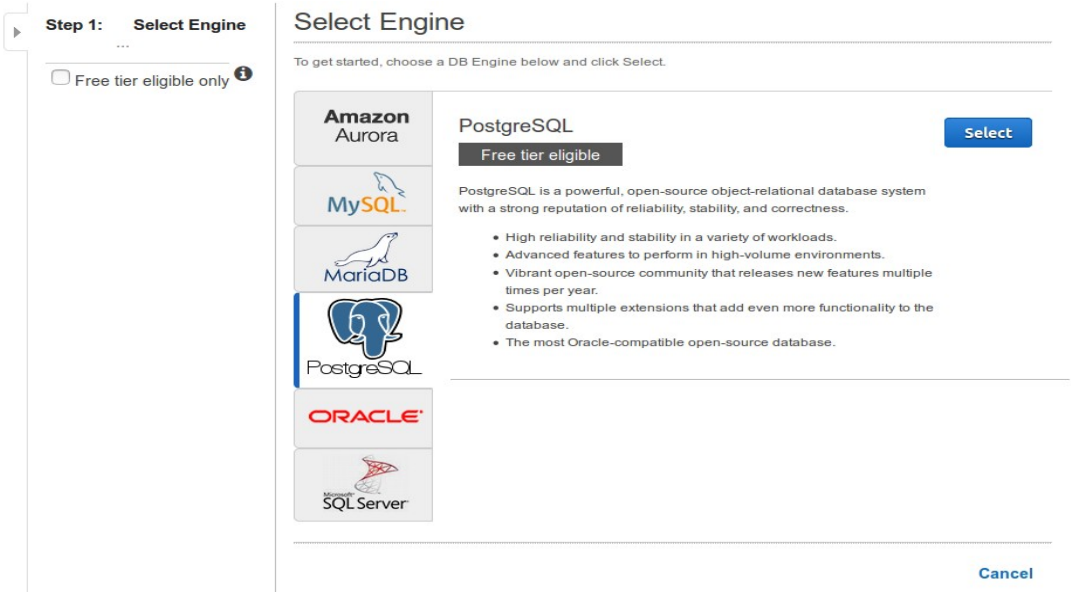
Amazon RDS es un servicio administrado de base de datos relacional, que nos permite levantar una base de datos en el entorno cloud de amazon web services https://aws.amazon.com/

Vamos a usar una instancia de base de datos Postgresql:

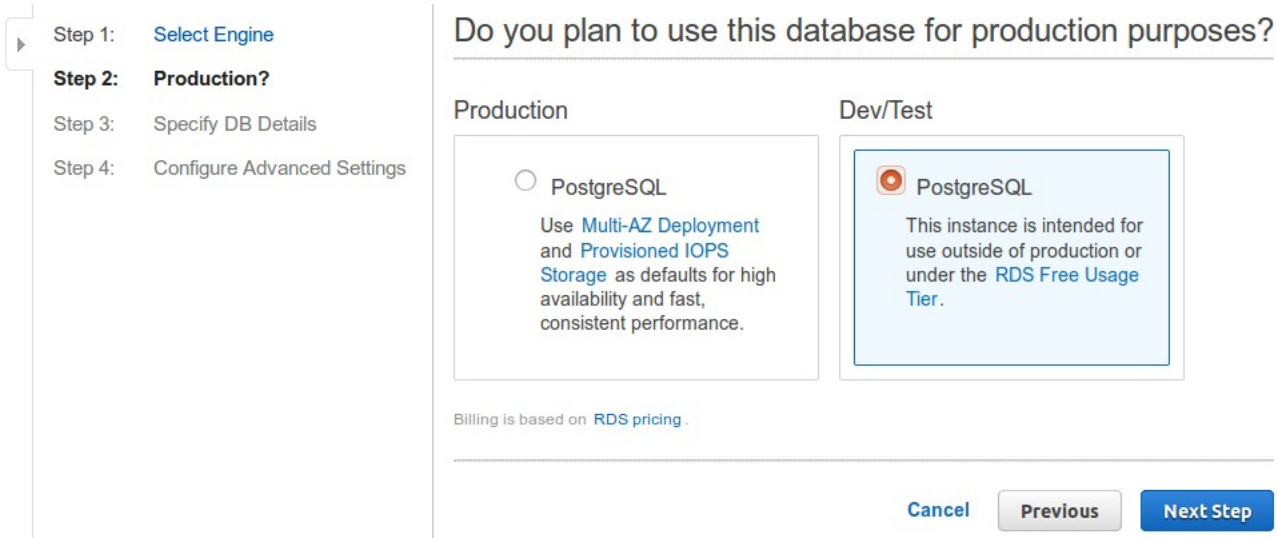
- Accedemos a nuestra cuenta de aws y seleccionamos el servicio RDS:



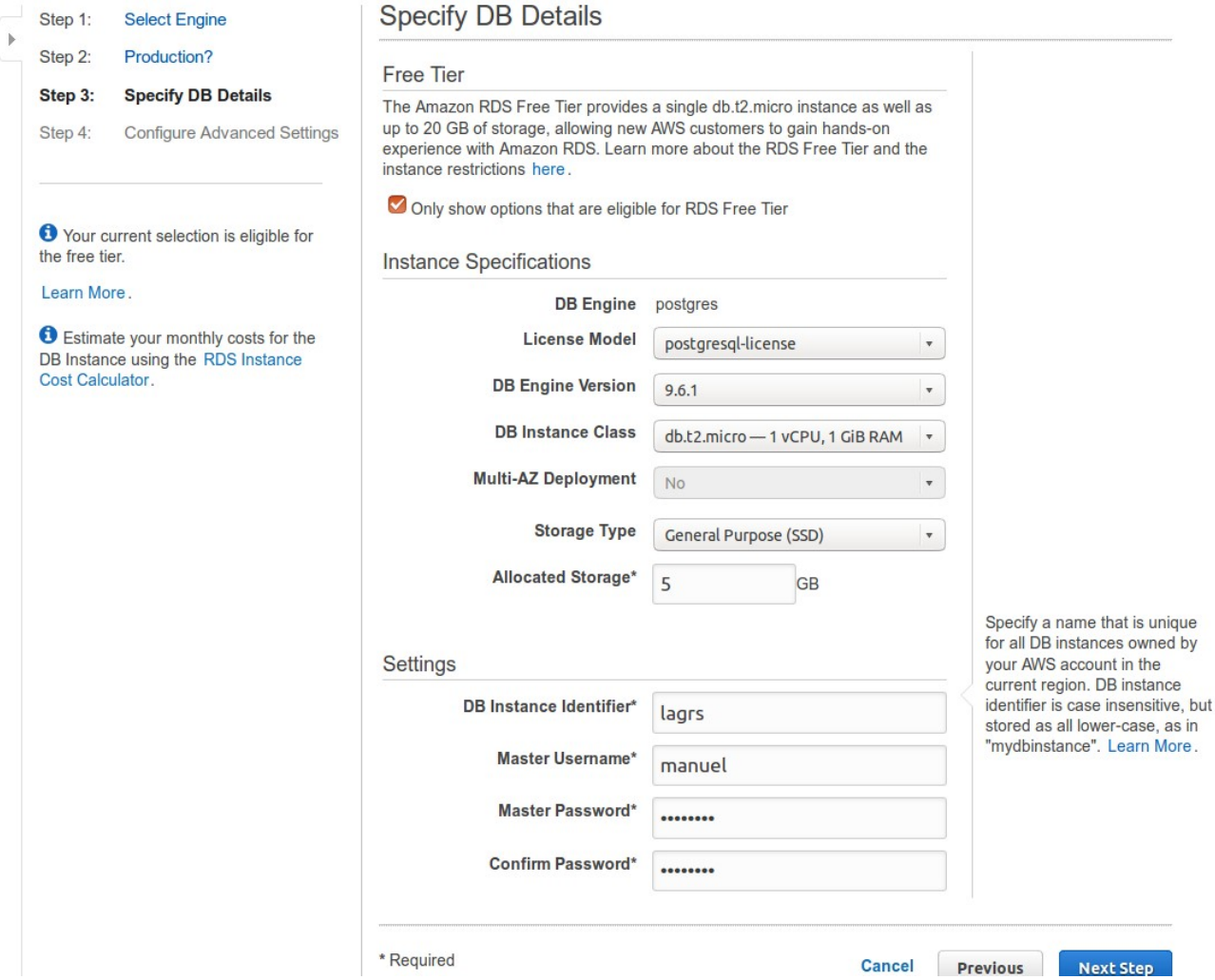
- En la RDS dashboard hacemos click en get started y selccionamos postgresql:



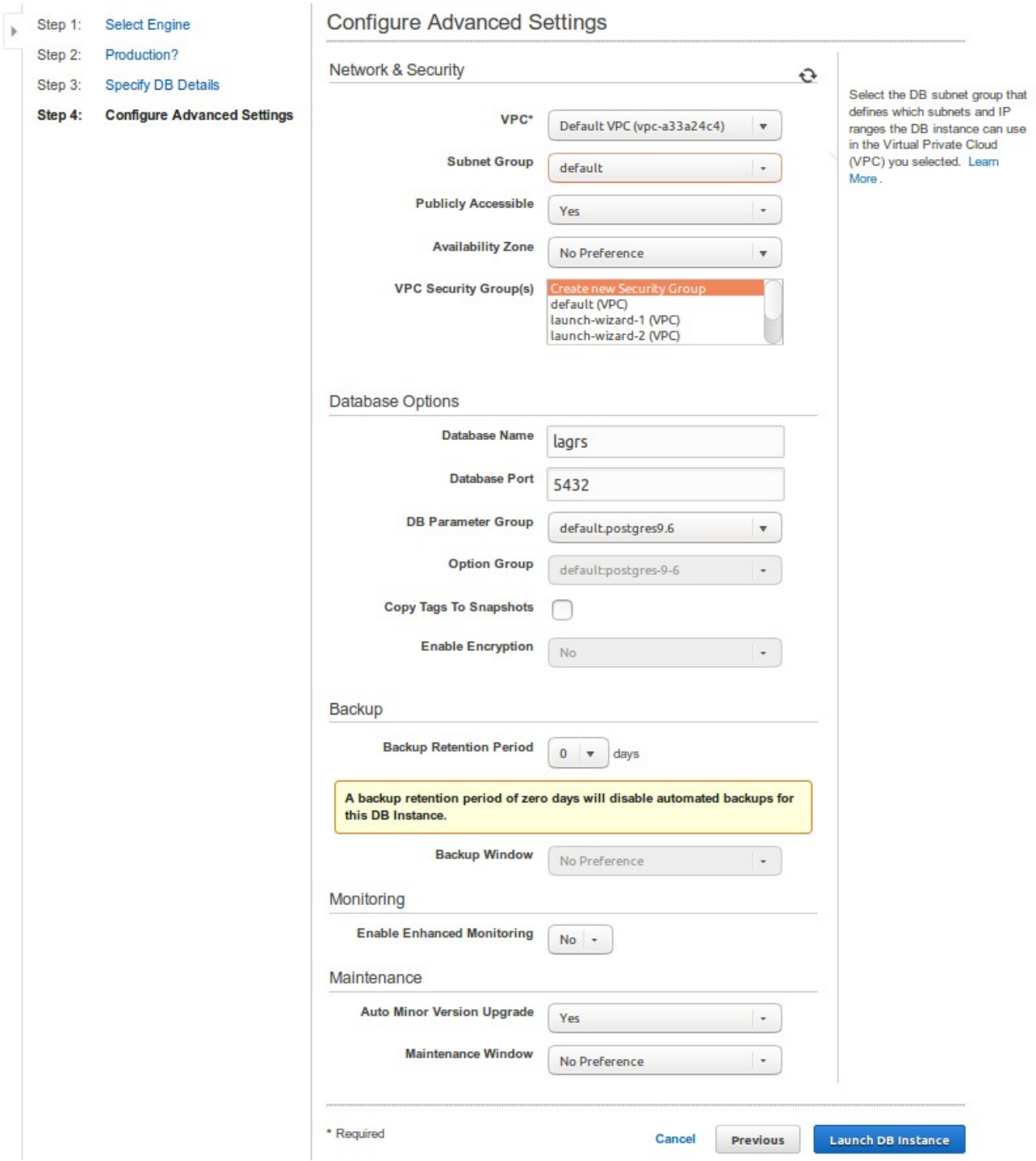
- Marcamos la opción Dev/Test para indicar que no la vamos a usar en producción:



- Ahora, especificamos los detalles de nuestra base de datos. Es importante marcar la opción “Only show options that are eligible for RDS Free Tier”:

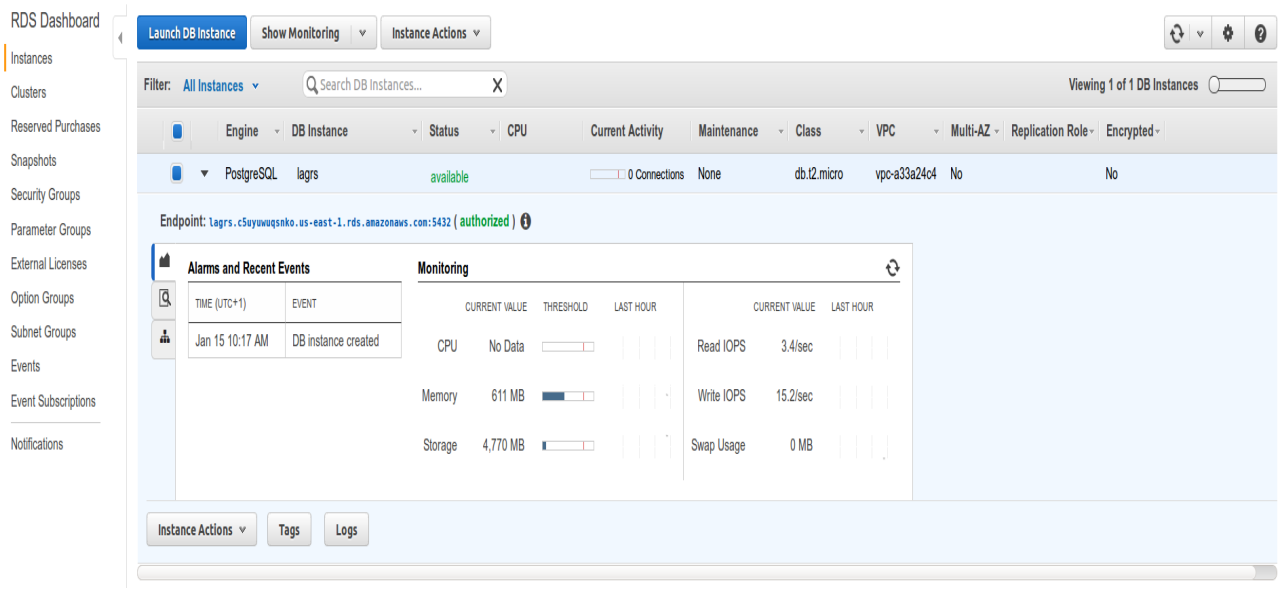


- Ahora configuramos las opciones avanzados. No cambiamos mucho, simplemente le damos nombre a la base de datos, desactivamos los backups porque no los necesitamos para la demostración y nos aseguramos de que es accesible públicamente. Dejamos el puerto por defecto para Postgresql, el 5432:



Lanzamos la instancia haciendo click en el botón “Launch DB Instance”. Tardará unos minutos en levantarse.

- Una vez creada, tendremos la consola de administración con información de la instancia:



Es importante fijarse en el Endpoint, ya que será la dirección pública para conectarse a la base de datos. Por ejemplo, en este caso es:

`lagrs.c5uyuwuqsnko.us-east-1.rds.amazonaws.com:5432`

Por lo que si quisiéramos conectarnos con el cliente psql lo haríamos:

```
[manuel@manubook ~]$ psql --host=lagrs.c5uyuwuqsnko.us-east-1.rds.amazonaws.com
--port=5432 --username=manuel --password --dbname=lagrs
Password for user manuel:
psql (9.6.1)
SSL connection (protocol: TLSv1.2, cipher: ECDHE-RSA-AES256-GCM-SHA384, bits:
256, compression: off)
Type "help" for help.

lagrs=> \da
                        List of aggregate functions
 Schema | Name | Result data type | Argument data types | Description
--------+------+------------------+---------------------+-------------
(0 rows)

lagrs=> \quit
```

Para eliminar la instancia hacemos click en el botón “Instance Actions” y seleccionamos “Delete”.

5. MINIKUBE

Minikube es un herramienta para crear en local un cluster orquestado con kubernetes. Hace uso de Virtual Box para crear un nodo donde poder experimentar con kubernetes. (https://github.com/kubernetes/minikube)

Es muy sencillo de usar ya que al instalarlo, se nos crea toda la configuración necesaria para el cluster en ~/.minikube. Aquí vamos a explicar algunos de los comandos que vamos a usar:

- start: levanta el cluster.

- stop: para el cluster.

- delete: borra la máquina virtual que se crea al levantar el cluster.

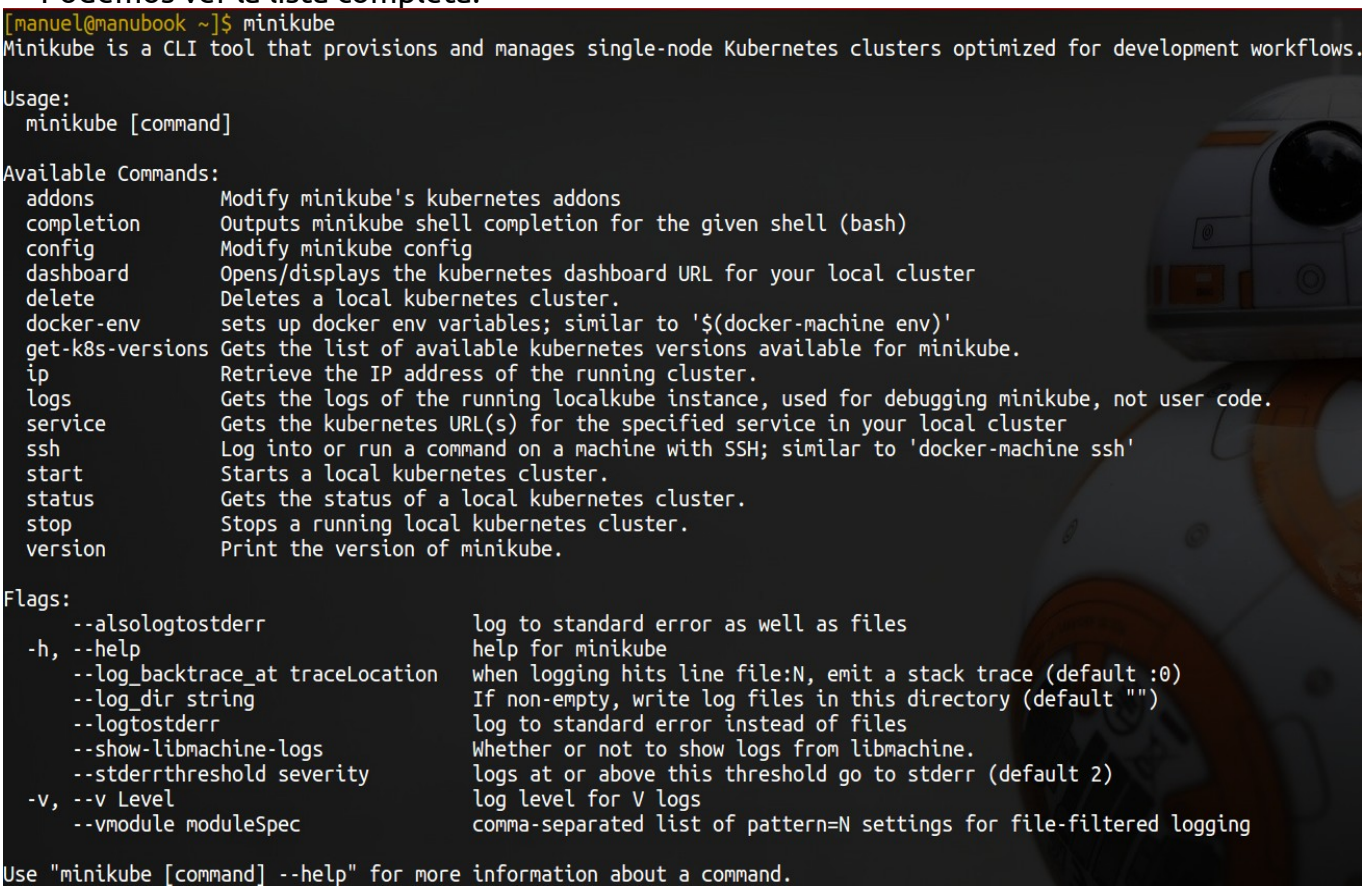
- addons: para gestionar los diferentes addons que se pueden usar con minikube.

- dashboard: nos lanza un panel web del cluster kubernetes donde podemos realizar todas las operaciones de manera gráfica.

- service: para averiguar la ip:puerto de algún servicio expuesto.

- ssh: para conectar con la máquina virtual.

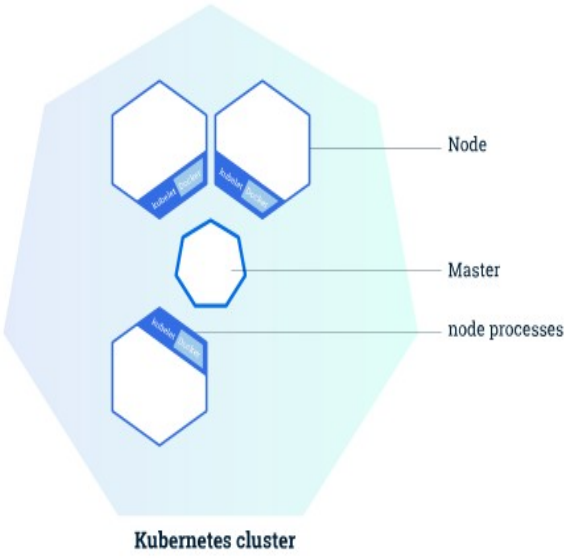
Podemos ver la lista completa:

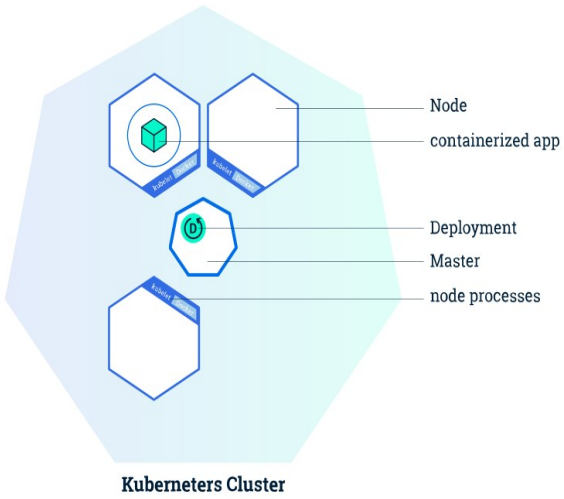


6. KUBERNETES

Kubernetes es un sistema open-source, desarrollado por Google, para la automatización del despliegue, escalado y administración de aplicaciones containerizadas. (https://kubernetes.io/)

Un cluster kubernetes consta de un nodo master que organiza y distribuye el trabajo entre los demás nodos.

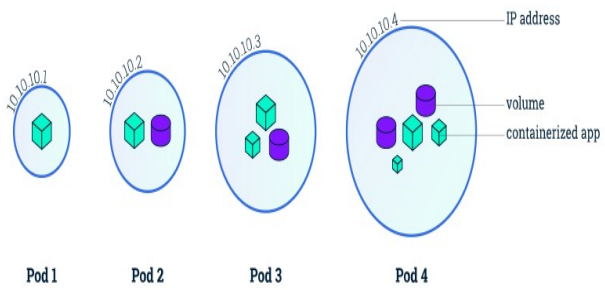


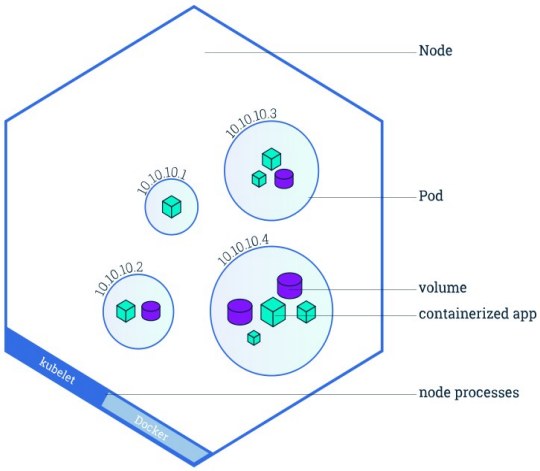


En cada nodo corre el servicio kubelet y un servicio de contenedores, el más extendido es Docker (https://www.docker.com/), aunque también se puede usar rkt (https://coreos.com/rkt/)

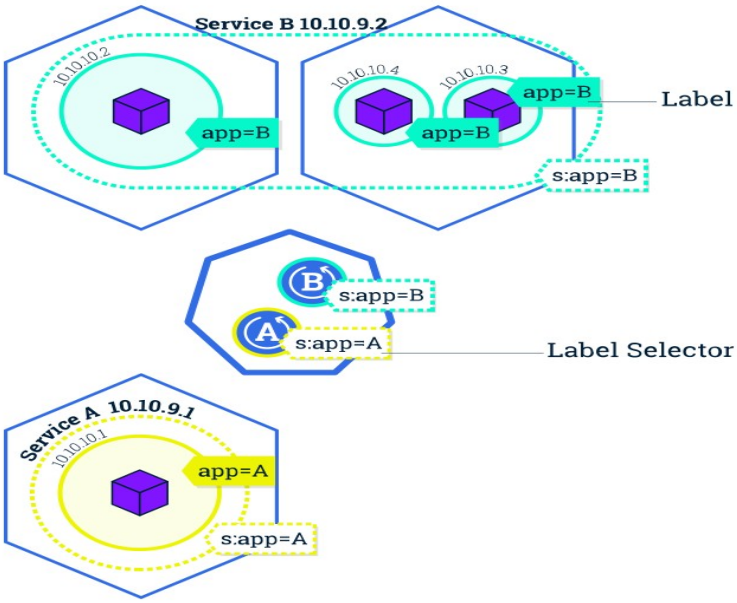
Kubernetes ofrece abstracciones para manejar los distintos elementos. Los objetos principales que nostros vamos a ver son:

- Pod: es la unidad mínima con la que se trabaja. Agrupa uno o más contenedores y/o volúmenes que se despliegan en un nodo.





- Servicio: agrupa varios uno o más pods que pueden estar distribuidos en varios nodos y los expone como unidad al resto del cluster.



- Deployment: es un proceso que corre en el nodo master y se encarga de checkear y mantener el estado de lor recursos desplegados (servicios, pods,...). Verifica el estado y lo compara con el estado deseado, si no coinciden realiza las operaciones oportunas para llegar al estado deseado (añadir o quitar replicas replicas, rearrancar pods, etc). Es un objeto que ofrece una abstracción mayor que otros objetos como Replica Controllers o Replicas Set.

- ConfigMap: configmap es un objeto que nos permite tener un almacén clave valor visible en todo el cluster y que puede ser consumido como variables de entorno en los contenedores (también de otras maneras, como ficheros en un volúmen montado).

- Secrets: es como un configmap que proporciona algo de seguridad. Los valores de las claves se almacenan en base64, aunque luego al consumirlos como variables de entorno son decodificados automáticamente.

Una de las muchas cosas buenas que tiene kubernetes, es que nos permite tener “clusters virtuales”, separando recursos por namespaces.

Para clasificar recursos dentro de un namespace, se le puede añadir a éstos “labels” que luego mediante “selectors” no permitirá obtener aquellos recursos que coincidan con nuestra búsqueda.

Para interactuar con el cluster, utilizaremos la herramienta kubectl, un comando que nos permitirá hacer todas las operaciones que deseemos.

Los principales comandos que usaremos son:

- kubectl get: lista recursos
- kubectl run: lanza un deployment para manejar pods

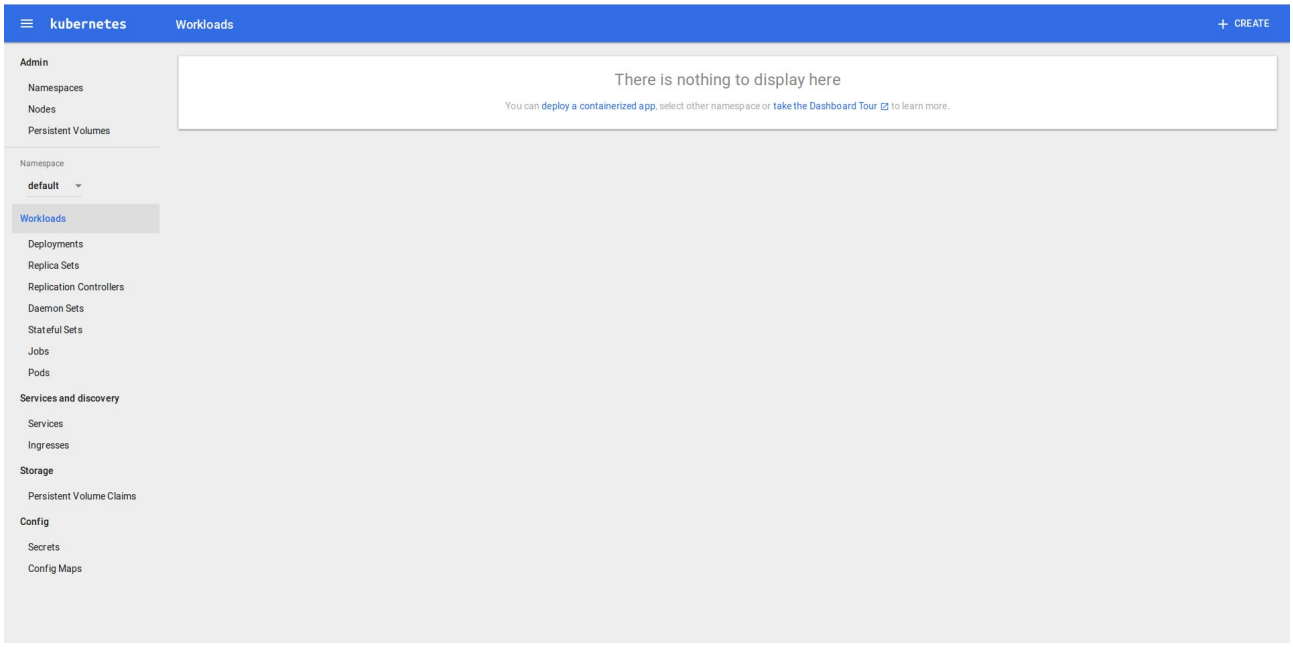
- kubectl expose: crea un servicio a partir de otro recurso
- kubectl exec: ejecuta un comando en un contenedor
- kubectl describe: información sobre un recurso
- kubectl logs: muestra los logs de un pod
- kubectl rolling-update: despliega la actualización de un recurso
- kubectl rollout: gestiona deployment rollout
- kubectl apply: Aplica una configuración a un recurso a partir de un fichero
- kubectl scale: escala un recurso
- kubectl autoscale: programa el escalado en función de unos parámetros
- kubectl delete: elemina recursos

La estructura de los comandos suele ser:
kubectl action resource options

Para obtener ayuda:
kubctl -h
kubectl action -h

Otra manera de interactuar con kubernetes es a través de un panel web, que en este caso podemos lanzar con el comando:

```
$ minikube dashboard
```



Para la realización del trabajo se trabará con la shell y se desplegarán los servicios mediante los archivos de configuración que se pueden encontrar en https://github.com/mcarmonaa/lagrs/tree/master/k8sconf