Arquitectura de Microservicios en Aws Usando un Cluster:Amazon Fargate usando Amazon EKS

Fernando . Barrrera[1], Diego . Chinchilla[2] y Miguel . Sanchez[3]

[1] *Escuela de ingenieria julio garavito, Bogota D.C , Colombia*

Fecha: 07/10/2020

Resumen— Este documento es con el fin de documentar nusetro proyecto de la materia de raquitecturas empresariales que tratara sobre el montaje de una arquietectura de microservicios en Aws usando principalmente el Amazon EKS y Amazon Forgate.

Palabras clave— AWS ,Micorservicios, Cluster , Docker,Nodo Master,Nodo Minion,EKS,Forgate,Escalamiento.

Abstract— This document is for the purpose of documenting our project on the subject of enterprise architecture that will deal with the assembly of a microservices architecture in Aws using mainly Amazon EKS and Amazon Forgate.

Keywords— AWS, Micorservices, Cluster, Docker, Master Node, Minion Node, EKS, Forgate, Scaling.

INTRODUCCION

Vamos a hablar de microservicios que corren en contenedores y son administrados por kubernetes, por lo cual discutiremos que son microservicios y como funciona kubernetes a un nivel general, con el objetivo de proveer al lector con conceptos basicos para que entienda mejor como funcionan los servicios de amazon EKS y fargate.

Hablando de Amazon EKS en pocas palabras es kubernetes pero en la nube lo dicutiremos mas adelan y con mayor profundidad, mientras que fargate es la implementacion de contenedores en la nube pero se debe tener una maquina on premise para administrarlos.

La intencion del parrafo anterior es para que el lector tengo una idea de sus diferencia principal aunque mas adelante entraremos mas en detalle sobre estas, el objetivo principal de en este infrome no va a ser discutir las diferencias entre los dos servicios y cual es mejor, es ver como sus diferencias pueden hacer que ambos servicios se complementen entre ellos.

ARQUITECTURA DE MICROSERVICIOS

Comúnmente el diseño de software se ha realizado de tal modo todos los aspectos funcionales quedan acoplados y sujetos en un mismo programa y servidor, esto generando acoplamiento, generando a largo plazo un problema en la escalabilidad. Gracias a ello aparece la arquitectura de microservicios, que nace al detectar la necesidad de tener un software más cambiante y una implementación rápida. La arquitectura de microservicios es un método de desarrollo de aplicaciones software la cual funciona como un conjunto de pequeños servicios que se ejecutan de manera independiente y autónoma, proporcionando una funcionalidad de negocio completa. Los microservicios se comunican entre sí a través de API's, y cuentan con sistemas de almacenamiento propios, lo que evita la sobrecarga y caída de la aplicación.(Decide, 2020)

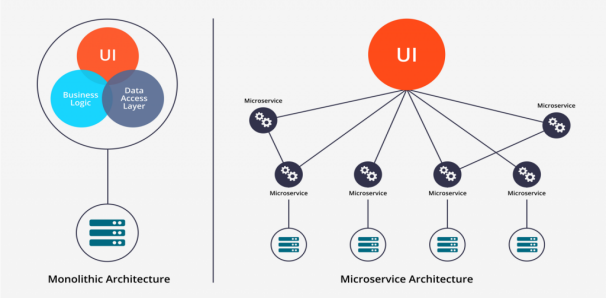

Fig. 1

Ventajas:

- Modularidad
- Escalabilidad
- Versatilidad
- Rapidez de actuación
- Mantenimiento simple y barato
- Agilidad

Desventajas:

- Alto consumo de memoria
- Inversión de tiempo inicial
- Complejidad en la gestión
- Perfil de desarrollador(experimentado)
- No uniformidad
- Dificultad en la realización de pruebas
- Coste de implantación alto

Cluster Kubernetes en Microservicios

Dado que la arquitectura de microservcios busca granularizar sus servicios de modo que cada servicio sea empaquetado y ejecutado sin tener que depender de un server principal para su despliegue,por esa razon lo Dockers se convierten en la forma ideal de desplegar los microservicios ,Puesto que el Docker encapsula el servicio junto con los minimos recursos para ejecutar el microservicio de esta forma se garantiza la independencia del servcio y su alta disponibilidad .(concatel, 2020)

Sin embargo aunque cada microservicios en un Docker es independientes,muchos de estos microservicios necsitaran comunicarse entre si para cumplir una funcion conjunta ,pero se complicaria bastante si lo Doscker se encontran en varios host o maquinas por lo cual se vuelve necesario tener un orquestador que nos ayude a gestionar los microservicios contenidos en dockers que pueden estar distribuidos en varios host o maquinas y este orquetador puede ser un Cluster Kubernetes.(concatel, 2020)

Un Cluster Kubernetes es un conjunto de maquinas virtuales que son denominadas nodos y cada nodo ejecuta aplicaciones que se encuentran contenidas en Dockers en un ambiente cloud,Basicamente un cluster se podria considerar un orquestador que se encarga de la gestion de los Dockers contenidos dentro de los nodos. (RedHat, 2020)(campusmvp, 2020)

Un cluster Kuberenetes basicamente se compone de almenos un nodo Master y nodos Minions,El nodo Master es el encargado de controlar el estado del cluster es decir es el encargado de controlar los estados de los nodos Minions del cluster,Ademas de decidir que nodo Minion va a ejecutar cada Docker y en que momento se ejecutan cada Docker.(campusmvp, 2020)

Por otro lado los nodos Mininos son lo que contienen los Dockers y por lo tanto son los que ejecutan las cargas de trabajo del cluster,Ya que cada nodo Minion puede tener uno o mas Pods que a su vez son los que contienen los Dockers que son los que realmente (campusmvp, 2020)

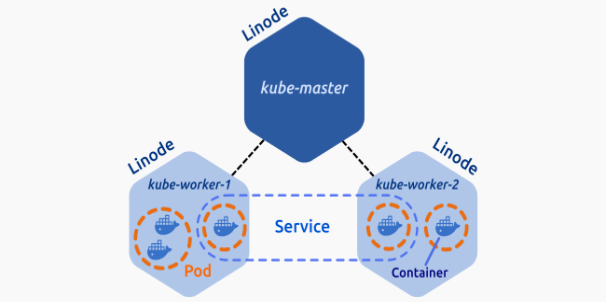

Fig. 2

Cada nodo Minion contiene Pods,Un Pod es un conjunto de Dockers los cuales pueden compartir recursos y comunicarse facilmente entre ellos y en caso de que los Dockers de un Pod necesite comunicarse con los Dockers de otro Pod que puede estar en otro nodo se hace uso de los servicios los cuales permiten la comunicacion de red entre Pods en un cluster Kubernetes como por ejemplo que en el Back-End de una aplicacion en java se encontrara en un Docker dentro del Pod 1 del Nodo 1 y el Front-End de la aplicacion en React puede estar en un Docker en el Pod 1 del Nodo 2 ,Entonces la forma por la cual el Front-End se comunicara con el Back-End sera haciendo uso de un servicio.(campusmvp, 2020)(RedHat, 2020)

Los clusters Kubernetes se adaptan a la perfeccion a las aplicaciones que implementen una arquitectuctura de microservicios,Ya que les facilita bastante la gestion y el monitoreo del conjunto de Dockers distribuidos que contienen los respectivos microservicios que componen la aplicacion general,Ademas de facilitar la escalabilidad horizontal y mejorar la disponibilidad ,Ya que en un cluster Kubernetes los Nodos pueden tener replicas en casos de que lo originales fallen y de esta forma no se ve afectado el flujo de trabajo.(concatel,

2020) (xataka, 2020)

AMAZON EKS

Amazon EKS o Amazon elastic kubernetes service es el servicio de ejecucion de kuberentes de Aws para automatizar la gestion,escalado e implementacion de las aplicaciones y servicios en Dockers,Amazon EKS pueden utilizar todos lo complementos y herramientas de kuberentes por lo que lo hace altamente compatible con cualquier aplicacion que se ejecute en kubernetes adicionalmente puede hacer otros servicios de Aws como Amazon IAM para la autenticacion,Amazon ECR para las imagenes de los contenedores ,Elastic Load Balancing para el balanceo de cargas de trabajo de los nodos del cluster. (AWS, 2020d)

Amazon EKS ofrece alta disponibilidad ya que ejecuta los recursos administrativos del clusters en varias zonas de disponibilidad ,puesto que localiza y remplaza los nodos del cluster que presenten errores ,otra gran ventaja de EKS es que ofrece la opcion de despliegue de servicios sin servidor mediante Amazon forgate,ademas mejora la seguridad del cluster mediante la instalacion automatica de parches de seguridad en los nodos del cluster.(AWS, 2020a)

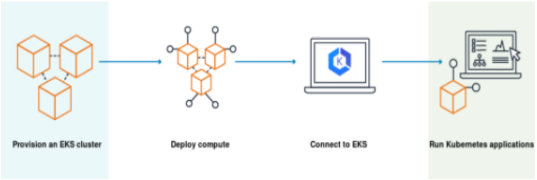

Fig. 3

Para el despligue de un cluster en Amazon EKS primeramente se crea el cluster desde la consola de AWS,despúes se procede a realizar el despliegue de los nodos minions administrados con instancias EC2 y nodos minions autoadministrables en serveless de Amazon Forgate,posteriormente se pueden configurar herramientas kubernetes para comunicarse con el cluster y gestionar el despliegue y ejecucion de las aplicaciones del cluster.(AWS, 2020d)

AMAZON FARGATE

AWS fargate es un motor de computo para contenedores sin servidor, lo cual hace que los desarrolladores no tengan que preocuparse por proveer o administrar los servidores que usen las aplicaciones ademas de que no se paga por maquinas creadas si no por recursos utilizados por la aplicaciones.(AWS, 2020b)

Fargate despliega y escala los contenedores con las especificaciones que tenga el contenedor, esto es posible gracias a que cada contenedor corre en su propio kernel y no comparten memoria, CPU, almacenamiento o red con otros contenedores.(AWS, 2020b)

Usar fargate es simple solo se tiene que construir el contenedor , definir los recursos que el contenedor va a usar y por ultimo correr y administrar las aplicaciones corriendo en los contenedores.(AWS, 2020b)

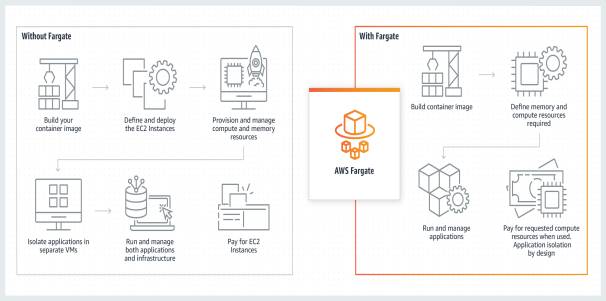

Fig. 4

AMAZON EKS VS AMAZON FARGATE

Ahora ya explicados ambos servicios de amazon, veremos sus ventajas y desventajas de la utilización de cada uno respecto al otro.

- EKS puede hacer uso de Fargate para el manejo de las aplicaciones. Esto quiere decir que Fargate se centra mas en las aplicaciones y no la infraestructura.
- Fargate permite especificar y pagar recursos por la aplicación. Por lo cual su precio es realmente flexible.
- EKS da la posibilidad de migrar fácilmente cualquier aplicación.
- EKS tiene alta disponibilidad ya que va cambiando los nodos que estén dañados, además de que posee muy buena seguridad.
- Fargate aisla la carga de trabajo ya que las tareas se ejecutan en su propio entorno. Por lo cual posee seguridad mediante aislamiento.
- Fargate Al asignar cantidad de computo se elimina la necesidad de elegir instancias y por ende el escalamiento no es necesario.

AMAZON FARGATE USANDO AMAZON EKS

Amazon EKS y fragate aunque estan diseñados para administrar contenedores tienen sus diferencias pero se complementa entre ellos por eso revisaremos los beneficios de utilizar ambos servicios al mismo tiempo.(AWS, 2020c)

Uno de los problemas con fargate es la diferencia principal con EKS y es que se paga por los recursos usados, si

por ejemplo tenemos una aplicacion que usa muchos recursos como CPU y RAM, fragate resultaria costosoy no seria la mejor opcion.(AWS, 2020c)

Por el lado de EKS si la aplicacion desplegada no usa muchos recursos y esta corriendo en una instancia EC2 no se estaria aprovechando todo el potencial de la instancia lo cual sigifica que se estaria pagando por recursos que nunca se usaron en todo su potencial.(AWS, 2020c)

Muchas de las aplicaciones actuales son complejas y hay servicios estas aplicaciones que pueden cosumir muchos mas recursos que otros al diferenciar estos microservicios y correr los que cosumen muchos recursos en instancias EC2 y aquellos que no consumen muchos recursos, estariamos aprovechando ambos servicios en todo su explendor y ahorrando dinero en el proceso.(AWS, 2020c)

ANALISIS EN DIFERENTES USOS

En esta sección veremos y analizaremos el comportamiento del uso de EKS y fargate por separado y otro análisis usando los dos juntos, compararemos principalmente los gatos por hora. En el eje Y veremos los gatos por hora resultantes de la aplicación y en el eje X el porcentaje de recursos usados por hora de la aplicación con los siguientes recursos para ambos 2 vCPU y 4 GB de RAM.

Usando EKS y fargate por separado para correr una aplicación de bajos requerimientos podemos ver que los gatos de usar únicamente instancias EC2 los gatos por hora es constante por lo que no importa que los recursos que pueda a llegar usar la aplicación no van a superar los recursos de la maquina virtual, fargate por el otro lado los contenedores usados por fargate son configurados para usar cierta cantidad de recursos los cuales serán cobrados por hora por lo cual en vemos que entre más recursos la aplicación usa mayores son los gastos.

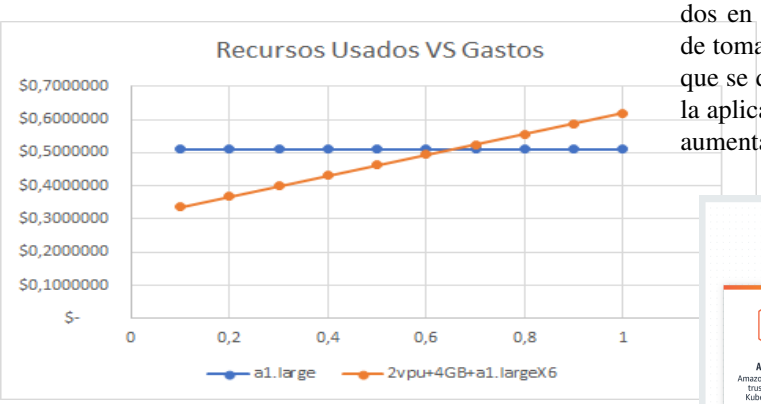

Fig. 5

Ahora usaremos una configuración diferencial de la misma aplicación, pero a una mayor escala de tal manera que usando únicamente instancias EC2 se deban usar 10 para un correcto funcionamiento y disponibilidad de la aplicación, mientras que usando ambos servicios juntos se usan 6 instancias EC2 y el resto de la aplicación corre en fargate, diferenciando los microservicios y los recursos usados por estos podemos disminuir los gatos y tener una mejor respuesta por parte los servicios corriendo en fargate y también la gran facilidad de auto escalamiento que tienen EKS y fargate cuando trabajan juntos.

Podemos ver que podemos disminuir los gatos mientras mejoras la funcionalidad de la aplicación, hay que aclarar de que se han dejado algunos gatos por fuero como el valor fijo de EKS por mes ya que afecta ambas formas de usar EKS o otros servicios recomendables como cloudwatch para logs y también algunas formas de disminuir costos como comprometiéndose a usar fargate con un uso mínimo del servicio y la reserva de máquinas EC2.

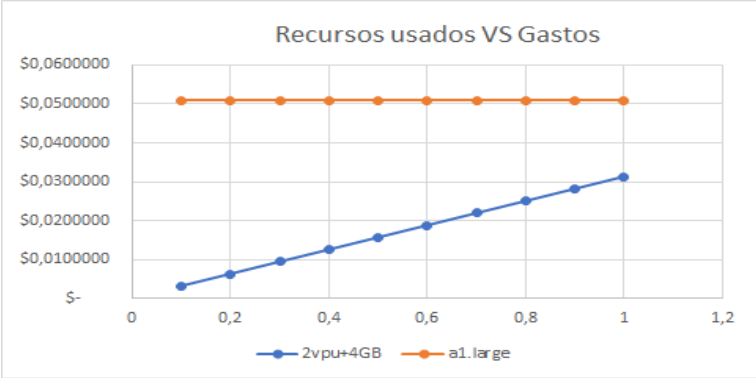

Fig. 6

Para concluir esta parte se debe tener en cuenta que se debería hacer un estudio y un conocimiento profundo de los servicios de AWS, Amazon tiene profesionales especializados en asesorar empresas en sus servicios, con el objetivo de tomar la mejor separación de microservicios de tal forma que se disminuyan los costos sin afectar la funcionalidad de la aplicación, en algunos caso pueden disminuir los gastos y aumentar la funcionalidad de la aplicación.

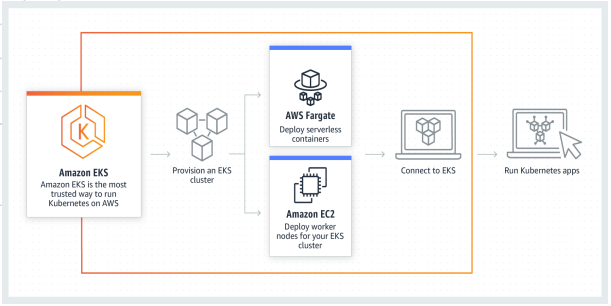

Fig. 7

MONTAJE DE ARQUITECTURA MICROSERVICIOS EN AWS

Para nuestro montaje de proyecto realizaremos la implementacion de un pequeño cluster en AWS haciendo uso del servicio Amazon Eks ,el Cluster contara con tres nodos trabajadores correspondiente a tres servicios de la aplicacion que seran el servicio de login ,servicio de chat y servicio de registro estos tres servicios seran deplegados en serveless de Amazon Forgate.

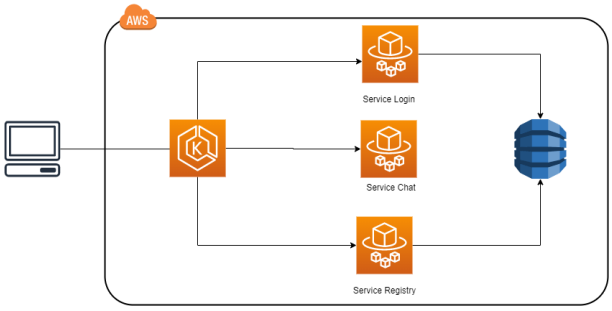

Fig. 8

Ademas se hara el depliegue de una base de datos DynamoDb para el almacenamiento de los usuarios y los mensajes resgistrados en el servcicio de chat de la aplicacion ,Basicamente al final se buscara que el cliente acceda al servicio login incialmente y de ahi pueda acceder al resto de los servicios del cluster.

PRUEBA DE CONCEPTO

Como se describio en la seccion anterior se hara la implementacion de un cluster kuberentes usando el servicio eks de aws con tres nodos faragte de las api de back y un nodo instacia EC2 para el front,para el despligue del cluster se creo una Virtual Private Cloud(VPC), la cual contaba con cuatro subredes ,dos subredes publicas y dos subredes privadas y en estas subredes sera donde realizaremos el despligue de los nodos del cluster.

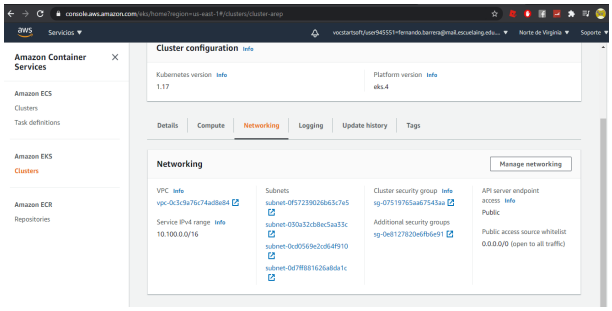

Fig. 9

Una vez se crea el cluster se procedera a la acrecaion de 3 nodos en fargata para las Apis y un nodo de tipo instancia EC2 para el fronto de la aplicacion,los tres nodos fargate se de desplegaran en la dos subredes privadas del cluster y el nodo instancia EC2 se desplegara en las subredes publicas y ahora veremos los nodos en en la panel del cluster de aws

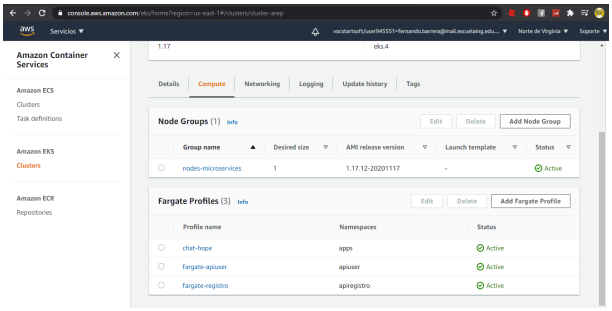

Fig. 10

Kubectl es una herramienta que nos ayudara con la comunicacion del cluster eks que hemos creado mediante esta herramienta podemos consultal los nodos del cluster y vemos los mimos 3 nodos que se muestran en el portal aws de cluster

```
C:\Users\jm_14\Downloads>kubectl get nodes
NAME                                       STATUS   ROLES    AGE    VERSION
fargate-ip-192-168-195-118.ec2.internal    Ready    <none>   2d4h   v1.17.9-eks-a84824
fargate-ip-192-168-214-124.ec2.internal    Ready    <none>   21h    v1.17.9-eks-a84824
fargate-ip-192-168-255-122.ec2.internal    Ready    <none>   21h    v1.17.9-eks-a84824
ip-192-168-18-103.ec2.internal             Ready    <none>   11h    v1.17.12-eks-7684af

C:\Users\jm_14\Downloads>
```

Fig. 11

Ahora consultaremos los namespaces del cluster y veremos que hay uno por cada servcicio uno para el servicio del chat ,otro para el servicio de la api de usuario y otro para la api de registro cada namespace se usa para el despligue de cada servicio de la arquitectura de microservicios del cluster

```
C:\Users\jm_14\Downloads>kubectl get ns
NAME              STATUS   AGE
apiregistro       Active   21h
apiuser           Active   21h
apps              Active   2d4h
default           Active   9d
kube-node-lease   Active   9d
kube-public       Active   9d
kube-system       Active   9d

C:\Users\jm_14\Downloads>
```

Fig. 12

Ahora veremso lo pods del cluster y veremos que por cada despligue de servcio tendermos un pod por lo cual veremso que los tres nodos fargate tendran su correspondiente pod ,por lo cual veremos un por para la api de usuarios,otro de la api de registros y otro correspondientes al chat como se vera acontinuacion

```
C:\Users\jm_14\Downloads>kubectl get pods -A
NAMESPACE     NAME                              READY   STATUS    RESTARTS   AGE
apiregistro   backregistro-7f6s74b4f-j6gs1      1/1     Running   0          21h
apiuser       backuser-89fbc4c55-cscxh          1/1     Running   0          21h
apps          chat-749b7b89d-xlb7b              1/1     Running   0          2d4h
kube-system   aws-node-7g4nn                    1/1     Running   0          11h
kube-system   coredns-75b44cb5b4-6qr8z          1/1     Running   0          11h
kube-system   coredns-75b44cb5b4-dm922          1/1     Running   0          11h
```

Fig. 13

Ahora se procedera a exponer los respectivos servicios y una vez lo servicio esten expuestos listaremos todos lo servicios del cluster y veremos un servicio de usuario,otro servicio de registro y otro servicio de chat como se vera acontinuacion

```
C:\Users\jm_14\Downloads>kubectl get service -A
NAMESPACE     NAME              TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)          AGE
apiregistro   service-registro  NodePort    10.100.122.187   <none>        8080:30825/TCP   19h
apiuser       service-user      NodePort    10.100.189.4     <none>        8080:32113/TCP   19h
apps          service-chat      NodePort    10.100.179.121   <none>        6677:30127/TCP   22h
default       kubernetes        ClusterIP   10.100.0.1       <none>        443/TCP          9d
kube-system   front             NodePort    10.100.187.54    <none>        3000:32140/TCP   12h
kube-system   kube-dns          ClusterIP   10.100.0.10      <none>        53/UDP,53/TCP    9d

C:\Users\jm_14\Downloads>
```

Fig. 14

Se procedera a realizara la descripcion de cada servicio donde se vera la IP del nodo fargate y el puerto por el cual esta corriendo cada servicio

- service-usuario

```
C:\Users\jm_14\Downloads>kubectl describe service  service-user -n apiuser
Name:                     service-user
Namespace:                apiuser
Labels:                   <none>
Annotations:              <none>
Selector:                 app=backuser
Type:                     NodePort
IP:                       10.100.189.4
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  32113/TCP
Endpoints:                192.168.214.124:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>

C:\Users\jm_14\Downloads>
```

Fig. 15

- service-registro

```
C:\Users\jm_14\Downloads>kubectl describe service  service-registro -n apiregistro
Name:                     service-registro
Namespace:                apiregistro
Labels:                   <none>
Annotations:              <none>
Selector:                 app=backregistro
Type:                     NodePort
IP:                       10.100.122.187
Port:                     <unset>  8080/TCP
TargetPort:               8080/TCP
NodePort:                 <unset>  30825/TCP
Endpoints:                192.168.255.122:8080
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>

C:\Users\jm_14\Downloads>
```

Fig. 16

- service-chat

```
C:\Users\jm_14\Downloads>kubectl describe service  service-chat -n apps
Name:                     service-chat
Namespace:                apps
Labels:                   <none>
Annotations:              <none>
Selector:                 app=chat
Type:                     NodePort
IP:                       10.100.179.121
Port:                     <unset>  6677/TCP
TargetPort:               6677/TCP
NodePort:                 <unset>  30127/TCP
Endpoints:                192.168.195.118:6677
Session Affinity:         None
External Traffic Policy:  Cluster
Events:                   <none>

C:\Users\jm_14\Downloads>
```

Fig. 17

Ahora procederemos a probar lo servicios de los 3 nodos fargate pero dado que estan desplegados en la VPC del cluster subredes privadas del cluster ,el unico que puede comunicarse con lo servicios de los tres nodos fargate es el nodo de la instancia EC2 del front que esta en la misma VPC del cluster y los nodos fargate pero en una subred publica ais que lo primero es encontrar la ip publica del nodo instancia EC2

```
  Kubelet is posting ready status
Addresses:
  InternalIP:   192.168.18.103
  ExternalIP:   3.81.174.81
  Hostname:     ip-192-168-18-103.ec2.internal
  InternalDNS:  ip-192-168-18-103.ec2.internal
  ExternalDNS:  ec2-3-81-174-81.compute-1.amazonaws.com
```

Fig. 18

Ahora se procede entrar por ssh al nodo intancia EC2 por medio de la ip publica del nodos intancia EC2 y probaremos la conexion a los respectivos servicios de los nodos fargate por medio de un curl y la ip privada de cada nodo faragate como se vera acontinuacion

```
ec2-user@ip-192-168-18-103:~
Last login: Fri Nov 27 17:24:12 2020 from 161.10.28.156

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
Last login: Fri Nov 27 17:24:12 2020 from 161.10.28.156

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-18-103 ~]$
```

Fig. 19

Ahora que ya ingresamos si procederemos al nodo intancia EC2 porcderemos a realizar la prueba de conexion a los 3 servicios de los 3 nodos fargate

- service-usuario

```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-192-168-18-103 ~]$ curl http://192.168.214.124:8080/users
[{"userName":"fernando","email":"fernando@mail.com","password":"arem123"}][ec2-user@ip-192-168-18-103 ~]$
```

Fig. 20

- service-registro

Fig. 21

- service-chat

Fig. 22

Con esto eveidenciamos que efectivamnete el nodo intancia EC2 se puede conectar con los tres servicios correspondientes a los 3 nodos fargate y dado que los 3 nodos faragate estan en una VPC el unico que puede acceder a ellos es el nodo instancia EC2 pero se podria conectar un datacenter a la VPC del cluster por medio de una coneccion vpn y de esta forma poder acceder a todos los nodos fargate ya que hasta el momento solo tenemos acceso al nodo EC2 que tiene acceso publico

Para finalizar se comprobo que se puede implementar una arquitectura de microservcio en un cluster kuberenetes con el servcio EKS de aws compuestos por nodos fargate y nodos instancias EC2 por lo cual se elige el tipo de los nodos dependiendo de la cantidad de recursos que necesite cada uno,ya que por nodos fargate se cobra por recursos utilizados y por nodos instancias EC2 se cobra por el tiempo que la maquina este prendida por lo cual si se necesitan muchos recurso es mejor una nodo instancia EC2 pero si el nodo no requiere tantos recurso es mejor un servelees fargate,por esa razon se puede optar o bien adoptar una arquitecturas de micoservcios usando serveless fargate,una arquitectura de microservcios con solo nodos EC2 o uno mixto con nodos fargate y nodos instancias EC2 todo depende de los recurso que necesiten los respectivos microservcios del cluster.

CONCLUSIONES

- Apesar de sus diferencias EKS tradicional y Amazon fargate pueden llegar a trabajar muy bien juntos y reducir costos de las aplicaciones corriendo en la nube.
- El uso apropiado de EKS y fargate juntos facilita la administracion de los contenedores no importa el tamaño del cluster.
- una arquitectura de microservicios se puede adaptar a una arquietectura serveless ,mediante la utilizacion de nodos serveless en el cluster

REFERENCIAS

[1] AWS (2020a). “Amazon elastic kubernetes service”. Tomado de https://aws.amazon.com/es/eks.

[2] AWS (2020b). “Aws fargate”. Tomado de https://aws.amazon.com/fargate.

[3] AWS (2020c). “Getting started with aws fargate using amazon eks”. Tomado de ttps://docs.aws.amazon.com/eks/latest/userguide/fargate-getting-started.html.

[4] AWS (2020d). “¿qué es amazon eks?” Tomado de https://docs.aws.amazon.com/es_es/eks/latest/userguide/what-is-eks.html.

[5] campusmvp (2020). “¿qué es kubernetes y cómo funciona?” Tomado de https://www.campusmvp.es/recursos/post/que-es-kubernetes-y-como-funciona.aspx.

[6] concatel (2020). “Microservicios, contenedores y kubernetes”. Tomado de https://sii-concatel.com/microservicios-contenedores-y-kubernetes.

[7] Decide (2020). “Microservicios”. Tomado de https://decidesoluciones.es/arquitectura-de-microservicios/?fbclid=IwAR3i9PDcLoSbIYwtMbs2PSlzb3Nl7b-Wvafq4AAQ71WvRj11X_aI5TYyckw.

[8] RedHat (2020). “Cluster kubernetes”. Tomado dehttps://www.redhat.com/es/topics/containers/what-is-a-kubernetes-cluster.

[9] xataka (2020). “De docker a kubernetes: entendiendo qué son los contenedores y por qué es una de las mayores revoluciones de la industria del desarrollo”. Tomado de https://www.xataka.com/otros/docker-a-kubernetes-entendiendo-que-contenedores-que-mayores-revoluc