IMPLEMENTACIÓN DE APLICACIONES EN CLÚSTER DE KUBERNETES DE AZURE Y DEMOSTRACIÓN DE SUPERVISIÓN Y MONITOREO

Pedroza Francisco Javier; Tobar Luis Felipe;
francisco.pedroza@uao.edu.co
luis.tobar@uao.edu.co
Facultad de Ingeniería
Universidad Autónoma de Occidente
Cali, Colombia

***Abstract* — In this project, an AKS (Azure Kubernetes Service) cluster with two nodes is deployed. Two different applications are configured on this cluster: the first contains an image classifier based on the Pytorch library and the second contains a website with a guestbook interface. This work seeks to understand the way in which the kubernetes cluster works and to be able to analyze the results obtained from the monitoring that Azure provides after mounting and making requirements to the applications that run on it. The different ways there are to work with this service are also tested, such as Cloud Shell from the same browser or with the Azure CLI, both from Windows PowerShell and through a Linux terminal.**

***Keywords* – AKS, cluster, nodes, Pytorch, guestbook, monitoring, Cloud Shell, Azure CLI .**

I. INTRODUCCIÓN

Azure Kubernetes Services(AKS) es un servicio de Microsoft de contenerización que nos permite de una manera sencilla implementar , administrar y usar Kubernetes, como un servicio de orquestador de contenedores, ofreciendo el uso y administración de clusters sobre máquinas virtuales y recursos de almacenamiento que deben ser pagos acorde a sus cualidades y tiempo de uso .
Desde su liberación de parte de Google en el año 2014 Kubernetes ha sido una plataforma portable y extensible para administrar servicios.
Kubernetes como sistema open-source para la automatización de despliegue, el escalado y la gestión de aplicaciones en contenedores permite hacer orquestación y aportar muchas más ventajas que otras soluciones en el mercado.

La orquestación de contenedores hace referencia a enlazar contenedores uno con otros permitiendo exponer servicios, escalar , organizar y desplegar aplicaciones sin afectar la disponibilidad del servicio de una manera automatizada en conjuntos de máquinas virtuales o cluster. Un cluster posee un estado que define las aplicaciones o cargas de trabajo a ejecutarse. el estado se define mediante un archivo de configuración(archivos tipo YAML) que indica el tipo de aplicación que se ejecutará y la cantidad de réplicas(escalamiento) para que un sistema funcione bien.

II. PLANTEAMIENTO DEL PROBLEMA

El reto que se propone cómo práctica es implementar un clúster de Kubernetes sobre el servicio de nube que ofrece Azure Portal de Microsoft. Anteriormente se realizó un clúster Kubernetes aprovisionando tres máquinas virtuales vagrant, y en éstas probando un par de aplicaciones para comprobar el funcionamiento. En este caso no se utilizarán archivos shell para aprovisionar las máquinas que se creen, sino que se tendrá que utilizar las herramientas que proporciona Azure Portal específicamente para crear clúster de Kubernetes, llamado AKS.
Como requerimientos se necesitará que dicho clúster cuente con por lo menos dos nodos y que se pueda comprobar el acceso a éste tanto por la Cloud Shell que ofrece el mismo Azure como a través de otra máquina utilizando el CLI de Azure.
Una vez se pueda trabajar sobre el clúster creado se debe implementar en el dos aplicaciones:

- La primera aplicación consiste en un clasificador de imágenes que utiliza la librería Pytorch para aplicar algoritmos de inteligencia artificial para cumplir dicha tarea.
- La segunda aplicación se propone por parte del grupo de trabajo, la cual es la implementación de una aplicación web de varios niveles, el cual consiste en un guestbook donde se busca que ambos integrantes del grupo pueda realizar anotaciones.

Ahora bien, a partir de las aplicaciones implementadas se debe probar el funcionamiento de los servicios de Monitoreo que ofrece el mismo Azure portal, donde se mostrará las métricas obtenidas referentes al clúster implementado y finalmente como punto extra se propone realizar un autoescalado en horizontal, donde se pueda tener flexibilidad en alguna aplicación.

II. DESCRIPCIÓN DE LA SOLUCIÓN

Para la primera fase que consiste en implementar el clúster de Kubernetes se utiliza la cuenta estudiantil de Azure con la cual se tiene acceso con el correo estudiantil de la Universidad Autónoma de Occidente, pero implica tener sólo 100 dólares de crédito para trabajar, lo que supone tener que destruir el clúster en los periodos donde no se vaya a trabajar.
Ahora bien, para crear dicho clúster se cuenta con dos opciones: se puede utilizar el Azure portal, que proporciona una interfaz web que despliega todas las configuraciones que se necesitan para lanzar el clúster, o utilizando el CLI de Azure, que por medio de comandos por terminal se pueden realizar también todas las configuraciones (en el caso de éste trabajo se realizó la implementación de ambas formas, pero se explicará detalladamente sólo con el CLI de Azure por medio de terminal, ya que a la hora de tener que montar varias veces el clúster, por el tema del límite de créditos, era más eficiente ejecutar unos cuantos comandos que llenar todos los apartados de la interfaz de Azure Portal).

Una vez se tiene el clúster implementado se va a proceder a implementar la aplicación de clasificación de imágenes con inteligencia artificial, utilizando un repositorio que se encuentra en github con el archivo yaml, el cual será modificado para utilizar la imagen propio que se tiene en docker hub cuando se probó dicha aplicación en las máquinas virtuales vagrant.

Posteriormente se sigue la guía de "Deploying PHP Guestbook application with redis" que proporciona la misma web de kubernetes, donde se contará con "un maestro de Redis de instancia única para almacenar entradas del libro de visitas, varias instancias de Redis replicadas para servir lecturas y Varias instancias de frontend web"[1]

Una vez se tiene lo anterior se puede realizar el monitoreo, donde se debe tener en cuenta que desde el momento de la creación del clúster se pudo activar la opción de poder hacer dicha acción, pero en caso de que no se haya hecho se podrá activar accediendo al cluster por terminal. Para entender todas la métricas que proporciona Azure portal se utilizarán las guías que proporciona la documentación del mismo Microsoft: "Monitoring a microservices architecture in Azure Kubernetes Service (AKS)"[2] y "Monitor your Kubernetes cluster performance with Azure Monitor for containers"[3].

Finalmente para realizar el punto extra de autoescalado en horizontal se propone seguir la guía que hay en la documentación de Kubernetes: "Horizontal Pod Autoscaler"[4] para intentar implementarlo con una de las aplicaciones.

III. PRUEBAS

Creación del clúster desde Azure portal:

Como se dijo en la descripción de la solución lo primero que se hace es la implementación del clúster. Desde la web de Azure portal se puede acceder al AKS (Azure Kubernetes Services) e iniciar a configurar todos los parámetros que se requieren para crearlo, como escoger o crear un grupo de recursos, definir la cantidad de nodos del clúster, especificar el tipo de máquina para los nodos, el nombre para el clúster, su región, la versión de kubernetes, determinar el prefijo para nombre DNS, habilitar enrutamiento por HTTP, si se desea o no la supervisión de los contenedores, entre otros.

Una vez creado se podrá ir a la opción de "conectar", donde se podrá obtener las credenciales para conectarse:

Figura 1: Comandos para acceder al clúster.

Una vez se tienen los comandos en la figura anterior se pueden ejecutar por ejemplo en la Cloud Shell, obteniendo luego de los dos comandos la siguiente salida:

```
Merged "k-master" as current context i
in /home/luis/.kube/confic
```

Ahora bien, para poder conectarse al clúster desde la PowerShell de Windows se necesita instalar el CLI de Azure con el siguiente comando:

```
Invoke-WebRequest -Uri https://aka.ms/installazurecliwindows -OutFile .\AzureCLI.msi; Start-Process msiexec.exe -Wait -ArgumentList '/I AzureCLI.msi /quiet'; rm .\AzureCLI.msi
```

Una vez se ha instalado se debe actualizar con "az update" para finalmente ya poder conectarse a la cuenta utilizando el comando "az login". Dicho comando abrirá en el navegador una página para loguearse (si se ejecuta en la terminal de la máquina local) y luego se ejecutan los comandos nuevamente de acceso al clúster:

```
PS C:\Users\Lenovo> az login
You have logged in. Now let us find all the subscriptions to
[
  {
    "cloudName": "AzureCloud",
    "homeTenantId": "693cbea0-4ef9-4254-8977-76e05cb5f556",
    "id": "bd5fc290-3803-458e-80f1-fd1ac4dd8b7d",
    "isDefault": true,
    "managedByTenants": [],
    "name": "Azure para estudiantes",
    "state": "Enabled",
    "tenantId": "693cbea0-4ef9-4254-8977-76e05cb5f556",
    "user": {
      "name": "luis.tobar@uao.edu.co",
      "type": "user"
    }
  }
]
PS C:\Users\Lenovo>
```

Figura 2: Ingreso a Azure por medio del CLI Azure.

Ahora bien, para poder visualizar los nodos, pods y servicios del clúster se debe instalar **kubectl.** En el caso de estar accediendo por terminal de Windows, se puede instalar kubectl utilizando Chocolatey, y una vez se haga esto se puede probar el acceso al clúster:

```
PS C:\Users\Lenovo> kubectl get services --all-namespaces
NAMESPACE              NAME
default                kubernetes
example                image-classifier
gatekeeper-system      gatekeeper-webhook-service
kube-system            addon-http-application-routing-default-http-backend
kube-system            addon-http-application-routing-nginx-ingress
kube-system            azure-policy-webhook-service
kube-system            calico-typha
kube-system            dashboard-metrics-scraper
kube-system            healthmodel-replicaset-service
kube-system            kube-dns
kube-system            kubernetes-dashboard
kube-system            metrics-server
PS C:\Users\Lenovo>
```

Figura 3: Comprobación de acceso a servicios del AKS por medio de Powershell desde máquina local.

Creación del clúster desde terminal con CLI Azure:

Para crear el clúster desde terminal se creó una máquina virtual vagrant con Ubuntu 18.04 LTS para poder utilizar comandos linux, ya que el equipo de trabajo los maneja mejor y también facilita seguir la documentación que se tiene. De este orden de ideas, se procede a levantar la máquina y a instalar el CLI de Azure para Linux, siguiendo los pasos de la instalación manual que se encuentra en al documentación

[1] Example: Deploying PHP Guestbook application with Redis. Disponible en línea en https://kubernetes.io/docs/tutorials/stateless-application/guestbook/

[2] Monitoring a microservices architecture in Azure Kubernetes Service (AKS). Disponible en línea: https://docs.microsoft.com/es-es/azure/architecture/microservices/logging-monitoring

[3] Monitor your Kubernetes cluster performance with Azure Monitor for containers Disponible en línea: https://docs.microsoft.com/es-es/azure/azure-monitor/insights/container-insights-analyze

[4] Horizontal Pod Autoscaler. Disponible en línea: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#:~:text=API%20Object,the%20autoscaling%2Fv1%20API%20version.

de Microsoft como “Install Azure CLI with apt”[5]. Una vez instalado se puede ejecutar “az login”, pero debido a que se está ejecutando en una máquina virtual no se va abrir la página de login en el navegador, sino que esto se debe hacer manual con la url y la clave que da como salida éste comando:

```
vagrant@maquina1:~$ az login
To sign in, use a web browser to open the page
https://microsoft.com/devicelogin and
enter the code RL2FJKFA9
```

Una vez se ha accedido se inicia con el proceso de crear el cluster (siguiendo el video tutorial “How To Create An Azure Kubernetes Service (AKS) Cluster Using Azure CLI And Cloud Shell” del canal TechSnips) iniciando con el grupo de cursos que va a contener al cluster con el siguiente comando:

```
vagrant@maquina1:~$ az group create
-n felipe-aks --location eastus
```

Con estos parámetros se le indica el nombre del grupo de recursos y la región que se desea (siendo este de Estados Unidos en este caso). Una vez se tiene el el grupo de recursos creado se pasa a crear el clúster dentro de éste:

```
az aks create --resource-group felipe-aks --name proyecto-cn
--node-count 2 --enable-addons monitoring
--generate-ssh-keys --kubernetes-version 1.19.3
--node-vm-size=Standard_B2s
```

Con el anterior comando se indica que dentro del grupo de recursos felipe-aks se cree el clúster con nombre proyecto-cn, al cual se le crearán dos nodos, su monitoreo estará activado, también que se desea que se genere las llaves SSH para conexión, se establece la última versión de kubernetes para el clúster y finalmente se escoge el tipo de máquina que se desea usar (En este caso se escogieron máquinas Standard_B2s debido que tenían el menor costo de créditos por mes).

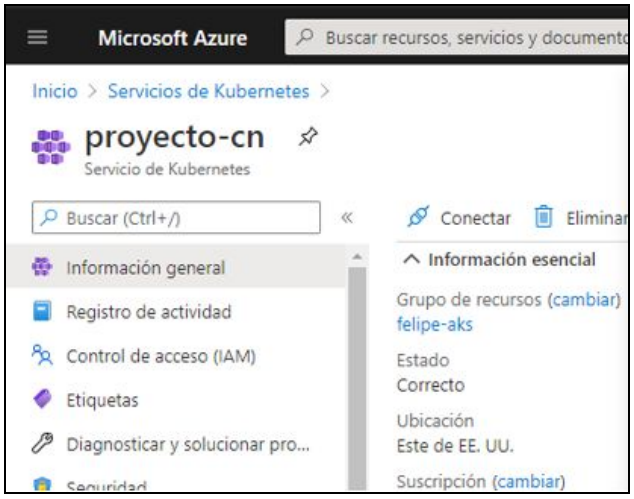


Figura 4: Comprobación de la creación del clúster desde Azure portal.

Implementación del clasificador de imágenes:

Para éste apartado se inicia creando una carpeta donde se clona el repositorio de github “pytorch-kubernetes”, el cual contiene el archivo de configuración image-classifier.yaml:

```
apiVersion: v1
kind: Namespace
metadata:
  name: example
  labels:
    name: staging
---
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: example
  labels:
    app: image-classifier
  name: image-classifier
spec:
  replicas: 1
  selector:
    matchLabels:
      app: image-classifier
  template:
    metadata:
      labels:
        app: image-classifier
    spec:
      containers:
      - image: felipetobars/image-classifier
        name: image-classifier
---
apiVersion: v1
kind: Service
metadata:
  namespace: example
  labels:
    app: image-classifier
  name: image-classifier
spec:
  ports:
  - port: 80
    protocol: TCP
    targetPort: 5000
  selector:
    app: image-classifier
  type: LoadBalancer
```

Figura 5: Archivo de configuración para el clasificador de imágenes.

Como se puede ver en la figura 5, los cambios diferentes respecto a la implementación pasada del clúster en las máquinas virtuales vagrant están en que la imagen que se va a utilizar es la propia que se cargó a docker hub en una práctica hecha previamente, y además para no tener que especificar la ip del nodo que contiene el servicio junto con el puerto (como se haría si el tipo de exposición del servicio fuera NodePort) se expone el servicio de forma “LoadBalancer”, lo que generará una ip externa que servirá como frontend para el servicio y es la que se usará para hacer requerimientos por parte de los usuarios. Para aplicar todas las configuraciones se ejecuta finalmente “**kubectl apply -f image-classifier.yaml**”, y se debe esperar a que el pod que se crea (se especificó sólo una réplica en el yaml) empiece a correr.

Implementación de servicio web-Guestbook:

A partir de la guía que proporciona kubernetes se puede montar el servicio siguiendo paso a paso los comandos que se indican en la documentación, sin embargo al ejecutar **kubectl apply** cómo se hace en dicha guía se define una url como ruta para buscar los archivos yaml. Al final esta misma guía se indica que esta forma en la que se hace es para exponer el servicio como NodePort, sin embargo dan la opción para hacerlo por LoadBalancer, pero para ello toca modificar un archivo yaml, por lo cual se creó una carpeta donde se tendrá cada archivo yaml que se necesita para implementar este servicio.

[5] Install Azure CLI with apt. Disponible en línea: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt

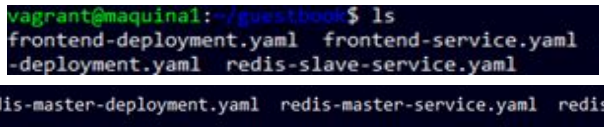


Figura 6: Archivos de configuración yaml para el servicio web de Guestbook.

El primer archivo de configuración que se aplica con kubeclt es el redis-master-deployment.yaml, que para esta implementación no se le hizo ninguna modificación respecto al que se muestra en la guía de kubernetes. En este archivo se define el pod redis-master, el cual se usará redis como app, su rol se define como master, la tier como backend y sólo se hace una réplica de éste. La imagen que se utiliza para esto es la que se encuentra en **k8s.grc.io/redis:e2e** y algo importante que sus recursos se limitan a **100m de cpu y 100Mi de memoria** (para un posible escalamiento en horizontal). También se define un puerto para el contenedor, siendo el 6379.

Figura 7: Confirmación de creación del pod de redis-master.

Una vez se tiene corriendo el pod del máster se procede a aplicar la configuración del redis-master-service.yaml, en el cual se definen tanto para los labels de metadata como para los argumentos de selector la app como redis, el rol como master y el tier tipo backend (también se define el puerto en el que se definió el master deployment, en el 6379, y ese mismo se define como targetPort) .

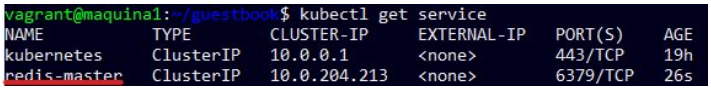


Figura 8: Confirmación de creación del servicio redis-master con puerto 6379.

Luego se hace el proceso de aplicar las configuraciones del redis-slave-deployment.yml, con configuraciones muy parecidas al master, cambiando el rol a slave, realizando 2 réplicas para éste, obteniendo la imagen desde **gcr.io/google_samples/gb-redisslave:v3,** u para obtener el nombre se utiliza el dns desde el host (también se limitan los recursos a 100m de cpu y 100Mi de memoria, y el puerto se define por 6379).

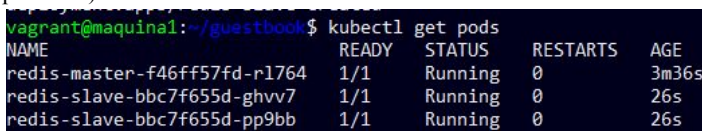


Figura 9: Confirmación de creación de dos réplicas del redis slave.

Ahora bien, para el redis-slave-service.yaml se utiliza nuevamente una configuración muy parecida al del redis máster, cambiando solamente el parámetro de "role" a slave.

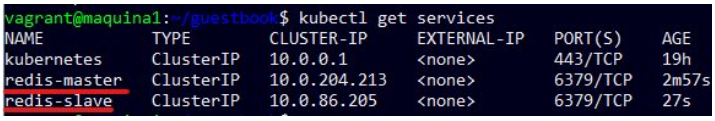


Figura 10: Confirmación de creación del servicio redis-slave con puerto 6379.

Finalmente se procede a "montar" el deployment y el service del frontend que se va a utilizar. En el archivo de frontend-deployment.yaml se define la "app" como guestbook, a diferencia de los anteriores archivos el tier también viene a ser frontend y no backend, se hacen tres réplicas, se obtiene la imagen a implementar desde **gcr.io/google-samples/gb-frontend:v4,** limitando sus recursos también a 100m de cpu y 100Mi de memoria, se define el nombre por medio de dns llamando al del host y se establece como puerto contenedor el 80.

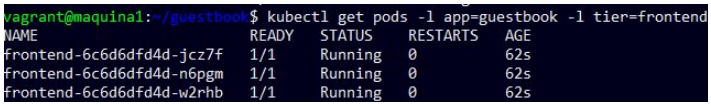


Figura 11: Confirmación de creación de tres réplicas del frontend.

Finalmente se pasa al archivo que se modifica, el cual es el frontend-service.yaml, donde se especifica que será un servicio, en labels se tendrá guestbook en app y frontend en tier, en spec se define el type como LoadBalancer (como se había explicado previamente) y se establece el puerto por el 80.

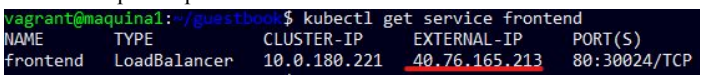


Figura 12: Confirmación de creación del servicio frontend.

Con todas las configuraciones anteriores ya se pueden probar las aplicaciones implementadas en el clúster y ver los resultados que se obtienen.

IV. RESULTADOS

Se realiza como primer despliegue la aplicación para la clasificación de imágenes ejecutando el archivo de configuración .yaml para crear y poner a funcionar el algoritmo. El servicio se despliega y se puede hacer uso del mismo mediante la ip externa que atenderá todas las solicitudes mediante una configuración de balanceo de carga propia del cluster AKS.

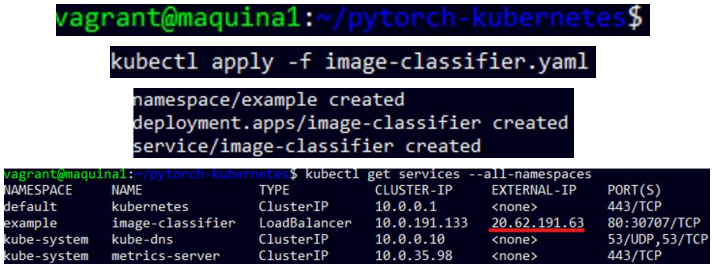
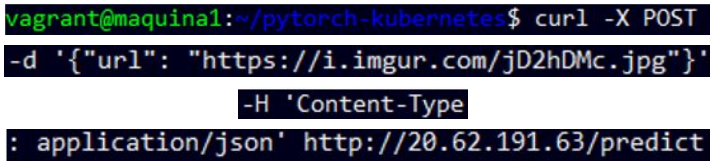


Figura 13: Creación de la aplicación contenerizada y visualización del servicio para clasificación de imágenes.

Posterior a la verificación del servicio en funcionamiento se procede a evidenciar mediante una petición de posteo al algoritmo de clasificación, donde enviamos la url de una imagen direccionada a la ip externa por medio de curl desde comandos y esperamos como respuesta la etiqueta asignada y un porcentaje de precisión.

Entrada:



Salida:

Figura 14: Puesta en marcha de la aplicación de clasificación de imágenes mediante petición curl.

Dentro del cluster implementado se despliega ahora la segunda aplicación la cual funciona a modo de formulario, ejemplo para escribir y publicar texto (posterior al despliegue de las diferentes capas de trabajo de la aplicación mediante servicios para el funcionamiento de backend y el frontend).

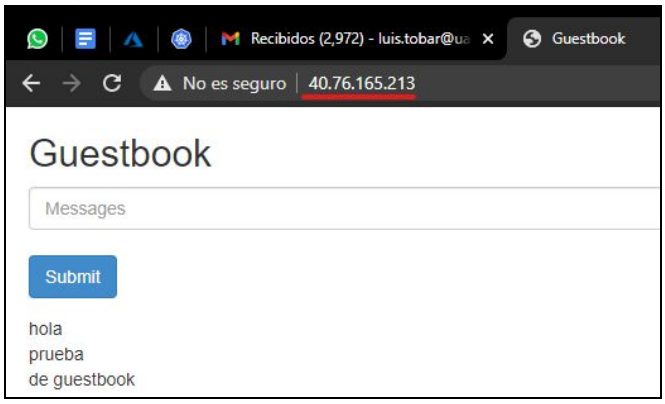


Figura 15: Aplicación Guestbook corriendo en el navegador con la ip externa de la figura 12.

Como se observa en la anterior figura, mediante la ip externa creada por el servicio para frontend, se puede acceder a la aplicación y realizar pruebas de funcionamiento mediante el navegador.

Ahora bien, durante la creación del cluster AKS se especificó que el cluster tuviera la opción de supervisión activado, con la finalidad de poder observar mediante la interfaz gráfica de Azure el comportamiento y uso de los componentes y recursos que conforman el cluster, así:

Figura 16: Azure monitor para supervisión del cluster implementado.

Como se ve en la anterior figura, Azure monitor permite visualizar el comportamiento del cluster implementado, dando un informe general de los nodos que lo componen junto con los pods actuales del usuario y del sistema, catalogando el cluster en un estado (que para esta implementación) es "healthy" debido a que el cluster no está presentando fallas en su funcionamiento.

Otro tipo de métricas que permite visualizar Azure respecto al clúster es el uso que se le está dando tanto a la CPU como a la memoria:

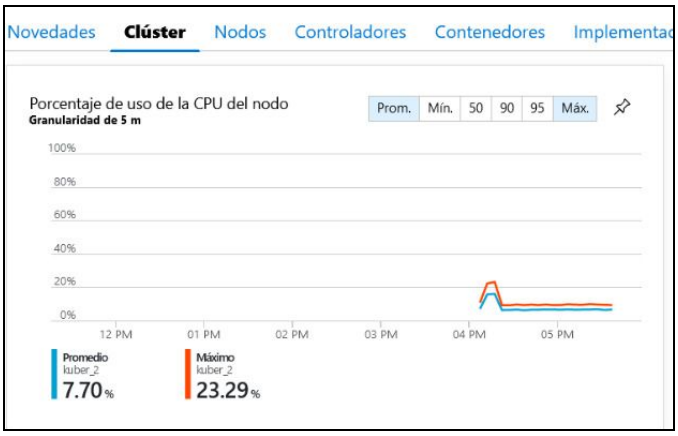


Figura 17: Azure monitor gráfica uso de CPU.

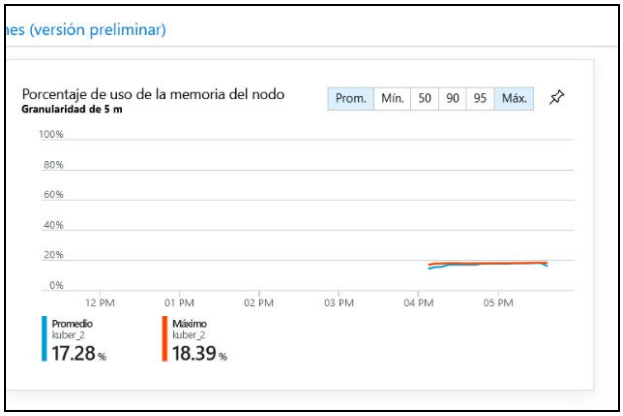


Figura 18: Azure monitor gráfica uso de memoria.

Como se ve en la figura 17 y 18 las máquinas virtuales escogidas para los nodos del clúster fueron suficientes para ejecutar los servicios implementados (el monitoreo que se presenta fué solamente con la aplicación de guestbook implementada).

Se observa en los gráficos que el consumo es mínimo debido a que las peticiones no fueron saturadas si no que se realizaron pocas pruebas por el tema de consumo de créditos en Azure.

Otro tipo de métricas que se pueden ver desde el Azure portal son relacionadas a los nodos, controladores y contenedores que pertenecen al clúster:

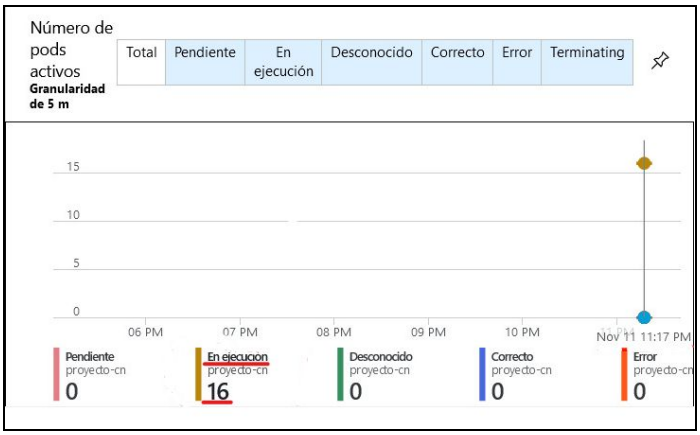


Figura 19: Azure monitor gráfica de análisis de pods.

Al crear el clúster se crean de forma predeterminada aproximadamente 10 pod, ya que esta era la cantidad que aparecía como nodos en ejecución antes de aplicar algún archivo de configuración para el Guestbook. Los pods agregados con ésta aplicación son 6: 1 réplica de redis-master, 2 réplicas de redis-slave y 3 réplicas del frontend, lo cual todo junto en total (los 10 pods por defecto y los 6 del guestbook) los 16 nodos en ejecución que entrega la gráfica de la figura 19.

Finalmente, de igual forma para comprobar cómo los controladores y contenedores (que se relacionan con los pods y servicios implementados) se distribuyen la carga de cpu y memoria que se visualiza en las figuras 17 y 18, se tiene un apartado en el Monitoring de Azure para visualizarlo:

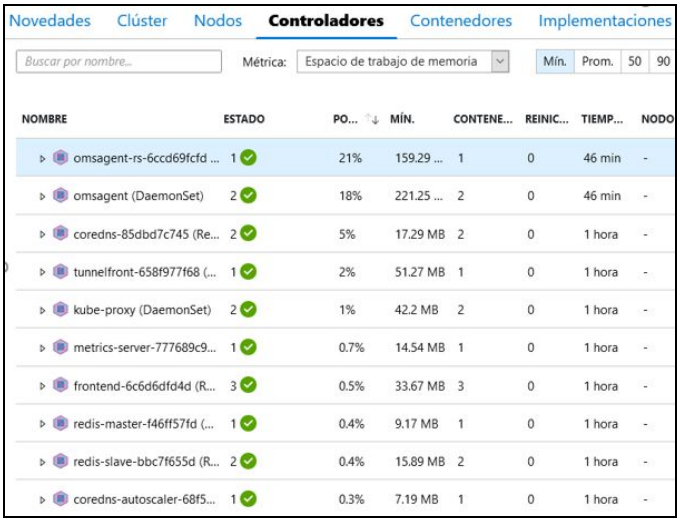


Figura 20: Azure monitor visualización de controladores.

Figura 21: Azure monitor visualización de contenedores.

V. CONCLUSIONES

- El uso de kubernetes para desplegar aplicaciones brinda modularidad de implementación como también administración de cargas de trabajo y servicios.
- El uso de clústeres y orquestadores permite escalar horizontalmente una aplicación y administrar todos los servicios desde un solo cluster, proporcionando más robustez ante fallas a los servicios que se proporcionen.
- AKS permite de una manera más simplificada administrar e implementar aplicaciones basadas en contenedores, esto gracias a todo su desarrollo de entorno en la nube mediante interfaces gráficas para implementar y seleccionar recursos y características, como también por el monitoreo y supervisión que proporciona.
- Azure monitor requiere menor esfuerzo técnico para su configuración lo que permite mayor eficiencia de supervisión y administración por su infraestructura visible, su sistema de notificaciones y programación automática de soluciones ante situaciones que puedan causar problemas como fallos. Además el hecho de que se brinde tanta documentación abre la posibilidad de que usuarios menos expertos puedan implementar sus aplicaciones, como también administrarlas, teniendo como soporte todas las herramientas que ya proporciona Azure.

REFERENCIAS

[1] Example: Deploying PHP Guestbook application with Redis. Disponible en línea en https://kubernetes.io/docs/tutorials/stateless-application/guestbook/. [Último acceso: 2020]

[2] Monitoring a microservices architecture in Azure Kubernetes Service (AKS). Disponible en línea: https://docs.microsoft.com/es-es/azure/architecture/microservices/logging-monitoring. [Último acceso: 2020]

[3] Monitor your Kubernetes cluster performance with Azure Monitor for containers Disponible en línea: https://docs.microsoft.com/es-es/azure/azure-monitor/insights/container-insights-analyze. [Último acceso: 2020]

[4] Horizontal Pod Autoscaler. Disponible en línea: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#:~:text=API%20Object,the%20autoscaling%2Fv1%20API%20version. [Último acceso: 2020]

[5] Install Azure CLI with apt. Disponible en línea: https://docs.microsoft.com/en-us/cli/azure/install-azure-cli-apt. [Último acceso: 2020]

[6] ¿Por qué todos apuestan por Kubernetes?. Paradigmadigital. Disponible en línea: https://www.paradigmadigital.com/techbiz/por-que-todos-apuestan-por-kubernetes/. [Último acceso: 2020]