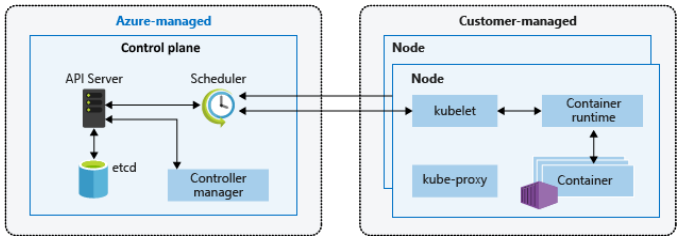
Informe Técnico del Microproyecto Numero 2 de Computación en la Nube – Kubernetes bajo Azure, clasificación de Imágenes y supervisión - monitoreo en Azure

Diana Patricia Delgado Paz
Email: diana_pat-delgado@uao.edu.co
Sergio Duván Mendoza
Email: sergio.mendoza@uao.edu.co

Se requiere la implementación de un clúster de Azure Kubernetes Service (AKS) de al menos dos nodos, mediante Azure Portal. Para ellos procedemos a establecer la arquitectura del clúster y su visualización gráfica:



I. INTRODUCCIÓN

Inicialmente realizamos una investigación de los conceptos estructurales de un cluster bajo kubernetes en Azure y encontramos la siguiente definición de conceptos:

Un clúster de Kubernetes se divide en dos componentes:

- Plano de control: proporciona los servicios básicos de Kubernetes y la orquestación de las cargas de trabajo de las aplicaciones.
- Nodos: ejecutan las cargas de trabajo de las aplicaciones.

II. ESPECIFICACIÓN DE REQUERIMIENTOS

1. Implementación de un cluster Kubernetes en Azure: Se debe crear un cluster con al menos 2 nodos y verificar su funcionamiento mediante Cloud Shell y CLI de Azure
2. Desplegar una aplicación de clasificación de imágenes en AKS y verificar su funcionamiento. Se sugirió el siguiente: https://opensource.com/article/20/9/deep-learning-model-kubernetes. Se requiere la descarga local de las imágenes.
3. Se debe desplegar una información de interés propio en Azure y verificar su funcionamiento.
4. Demostrar el uso de los servicios de supervisión y monitoreo que provee AKS con alguna de las aplicaciones desplegadas en el clúster.

III. DESARROLLO DEL PROYECTO

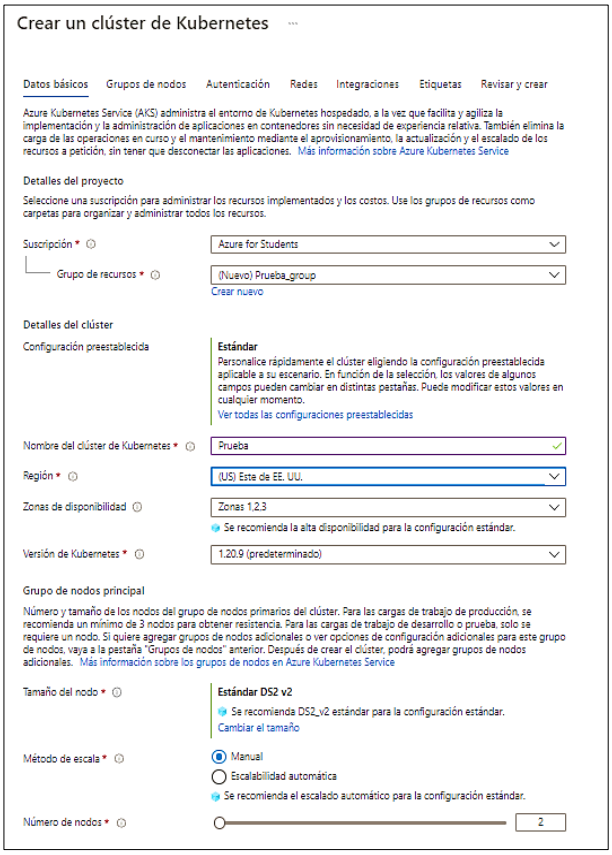
A. Implementacion del cluster Kubernetes:

en la cuenta de Azure se debe seleccionar la creación de un clúster y especificar lo siguiente:

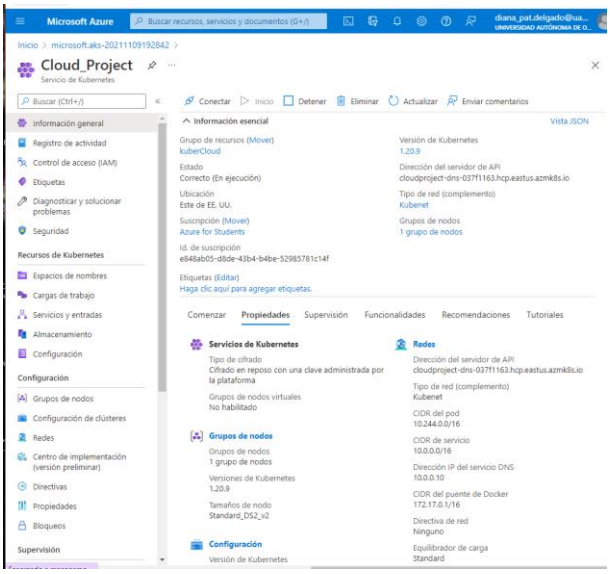
Especificar para el caso de nuestr cuenta de estudiante utilizamos los siguientes valores:

Nombre del grupo de recursos: kuberCloud
Nombre del clúster: Cloud_Project
Región: Este de USA
Tamaño del nodo: estándar.
Método de escala: manual
Numero de nodos:2

Documento realizado en octubre del año 2021. Este se realiza con base en la información del curso de Computación en la Nube dictado por el docente Oscar Mondragón de la universidad Autónoma de Occidente, como contenido de la especialización en Inteligencia Artificial
.



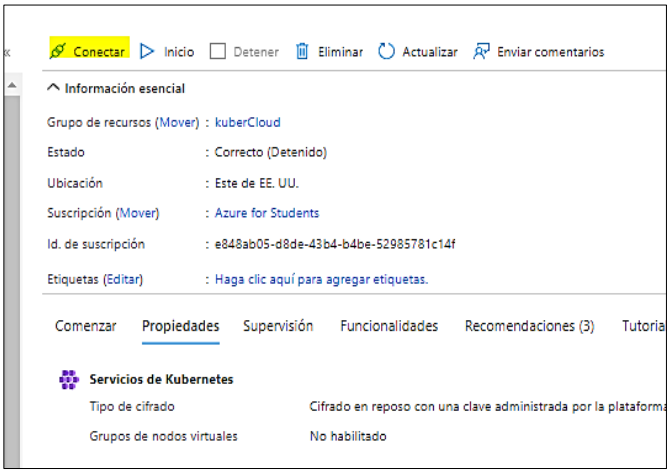
Seguidamente nos aparece la confirmación y configuración inicial del clúster:



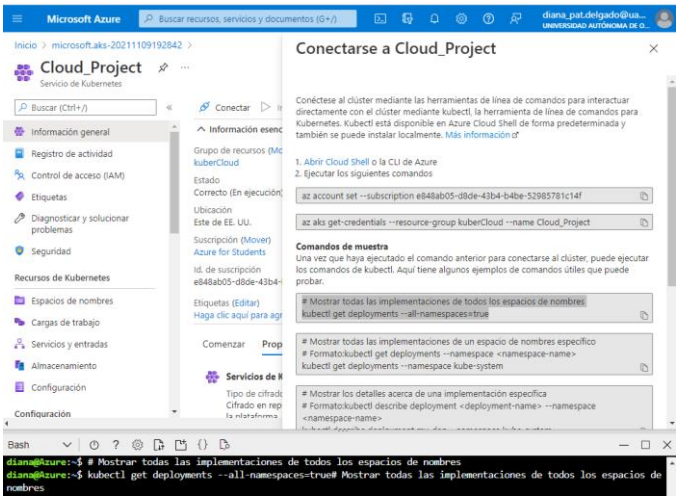
B. Verificar funcionamiento del clúster"

En este paso debemos esperar a que finalice la creación del clúster, seguidamente debemos conectarnos a la consola de Azure para validar por comandos el funcionamiento del mismo:

- Seleccionamos la opción de conexión e ingresamos los comandos que aparecen en la ventana emergente para loguearnos en el clúster.



Seguidamente aparece una sección en la parte inferior de la pantalla con la consola:



Realizamos la verificación del clúster a través de la revisión d ellos servicios que se están ejecutando:

- kubectl get ns

```
diana@Azure:~$ kubectl get ns
NAME              STATUS   AGE
default           Active   2d3h
example           Active   2d3h
kube-node-lease   Active   2d3h
kube-public       Active   2d3h
kube-system       Active   2d3h
```

- kubectl get services --all-namespaces

```
diana@Azure:~/pytorch-kubernetes$ kubectl get services --all-namespaces
NAMESPACE     NAME                          TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)         AGE
default       kubernetes                    ClusterIP   10.0.0.1       <none>        443/TCP         54m
example       image-classifier              NodePort    10.0.136.88    <none>        80:30347/TCP    3m48s
kube-system   healthmodel-replicaset-service ClusterIP  10.0.109.185   <none>        25227/TCP       53m
kube-system   kube-dns                      ClusterIP   10.0.0.10      <none>        53/UDP,53/TCP   53m
kube-system   metrics-server                ClusterIP   10.0.205.94    <none>        443/TCP         53m
diana@Azure:~/pytorch-kubernetes$ curl -X POST -d '{"url": "https://i.imgur.com/jD2hDMc.jpg"}' -H 'Content-Type:
```

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kubermatic-dl-deployment
spec:
  selector:
    matchLabels:
      app: kubermatic-dl
  replicas: 3
  template:
    metadata:
      labels:
        app: kubermatic-dl
    spec:
     containers:
     - name: kubermatic-dl
       image: kubermatic00/kubermatic-dl:latest
       imagePullPolicy: Always
       ports:
       - containerPort: 8080
```

Se ejecuta el deployment con el comando:

Kubectl apply -f deployment.yaml

Se verifica el resultado de la clasificación de una imagen, la cual previamente ha sido descargada de la web con el comendo wget y validamos corriendo el comando curl, en el cual se especifica el nombre de la imagen y la ip del nodo que contiene el clasificador de imágenes:

```
diana@Azure:~/images$ wget https://i.imgur.com/jD2hDMc.jpg
--2021-11-10 02:08:16--  https://i.imgur.com/jD2hDMc.jpg
Resolving i.imgur.com (i.imgur.com)... 151.101.48.193
Connecting to i.imgur.com (i.imgur.com)|151.101.48.193|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 39300 (38K) [image/jpeg]
Saving to: 'jD2hDMc.jpg'

jD2hDMc.jpg                 100%[=====================================================>]  38.38K  --.-KB/s    in 0.001s

2021-11-10 02:08:16 (34.4 MB/s) - 'jD2hDMc.jpg' saved [39300/39300]

diana@Azure:~/images$ ls -la
total 64
drwxr-xr-x 2 diana diana  4096 Nov 10 02:08 .
drwxr-xr-x 8 diana diana  4096 Nov 10 02:08 ..
-rw-r--r-- 1 diana diana   397 Nov 10 01:34 deployment.yaml
-rw------- 1 diana diana 12288 Nov 10 01:34 .deployment.yaml.swp
-rw-r--r-- 1 diana diana 39300 Jul 29  2013 jD2hDMc.jpg
diana@Azure:~/images$ curl -X POST -F img=@jD2hDMc.jpg http://20.81.95.232/predict
The input picture is classified as [cat], with probability 0.995.diana@Azure:~/images$
```

curl -X POST -F img=@horse.jpg http://10.96.104.184/predict

IV. DESPLIEGUE DE UN CLASIFICADOR DE IMÁGENES

Se realiza descarga desde la pagina sugerida del clasificador de imágenes para despliegue, se realiza creación de una carpeta “images” para desplegar la API:

- mkdir images

se realiza la creación del archivo deployment.yaml y se realiza copia del siguiente código:

```
diana@Azure:~/images$ kubectl get services
NAME                      TYPE           CLUSTER-IP     EXTERNAL-IP    PORT(S)          AGE
kubermatic-dl-deployment  LoadBalancer   10.0.240.150   20.81.95.232   80:30538/TCP     2d17h
kubernetes                ClusterIP      10.0.0.1       <none>         443/TCP          2d18h
diana@Azure:~/images$ curl -X POST -F img=@horse.jpg http://20.81.95.232/predict
The input picture is classified as [horse], with probability 0.864.diana@Azure:~/images$
diana@Azure:~/images$ curl -X POST -F img=@jD2hDMc.jpg http://20.81.95.232/predict
The input picture is classified as [cat], with probability 0.995.diana@Azure:~/images$
```

Según la imagen se verifica que se ejecutó la clasificación de la imagen de un gato con un porcentaje de probabilidad del 99.5 %

```
The input picture is classified as [cat], with probability 0.995.
```

Y para el caso de la imagen del caballo se presenta un porcentaje de probabilidad del 86.4%

V. EJECUTAR UNA APLICACIÓN DE INTERES

Se seleccionó para este ejercicio de despliegue una aplicación de stateless, en la cual se específica el siguiente archivo yaml:

```yaml
apiVersion: apps/v1 # Usa apps/v1beta2 para versiones anteriores a 1.9.0
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2 # indica al controlador que ejecute 2 pods
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

Seguidamente se ejecuta el despliegue de la aplicación y se verifica la creación de las replicas, ya que en le archivo se define 1:

```
diana@Azure:~/nginx$ kubectl apply -f nginx-deployment.yaml
Warning: resource deployments/nginx-deployment is missing the kubectl.kuberne
bectl apply. The missing annotation will be patched automatically.
deployment.apps/nginx-deployment configured
diana@Azure:~/nginx$ kubectl get pods -l app=nginx
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5d59d67564-kwpwk   1/1     Running   0          4m21s
nginx-deployment-5d59d67564-qbfjw   1/1     Running   0          4m15s
```

Verificamos uno de los pods:

```
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-5d59d67564-kwpwk   1/1     Running   0          9m52s
nginx-deployment-5d59d67564-qbfjw   1/1     Running   0          9m46s
diana@Azure:~/nginx$ kubectl describe pod nginx-deployment-5d59d67564-qbfjw
Name:         nginx-deployment-5d59d67564-qbfjw
Namespace:    default
Priority:     0
Node:         aks-agentpool-70570969-vmss000006/10.240.0.4
Start Time:   Fri, 12 Nov 2021 20:29:59 +0000
Labels:       app=nginx
              pod-template-hash=5d59d67564
Annotations:  <none>
Status:       Running
IP:           10.244.1.4
IPs:
  IP:           10.244.1.4
Controlled By:  ReplicaSet/nginx-deployment-5d59d67564
```

```
Containers:
  nginx:
    Container ID:   containerd://5459b0619b856a0ac4826065b4a6e44faa09248dcb4ee7d396086fdc1fedd0b2
    Image:          nginx:1.7.9
    Image ID:       sha256:35d28df486f6150fa3174367499d1eb01f22f5a410afe4b9581ac0e0e58b3eaf
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 12 Nov 2021 20:30:00 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-cx7tb (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-cx7tb:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-cx7tb
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
```

Verificamos el listado de eventos asociados al pod:

```
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  10m   default-scheduler  Successfully assigned default/nginx-deployment-5d59d67564-qbfjw to aks-agentpool-70570969-vmss000006
  Normal  Pulled     10m   kubelet            Container image "nginx:1.7.9" already present on machine
  Normal  Created    10m   kubelet            Created container nginx
  Normal  Started    10m   kubelet            Started container nginx
```

Seguidamente hacemos el ejercicio de actualización del deployment para que genere 2 replicas y verificamos el cambio:

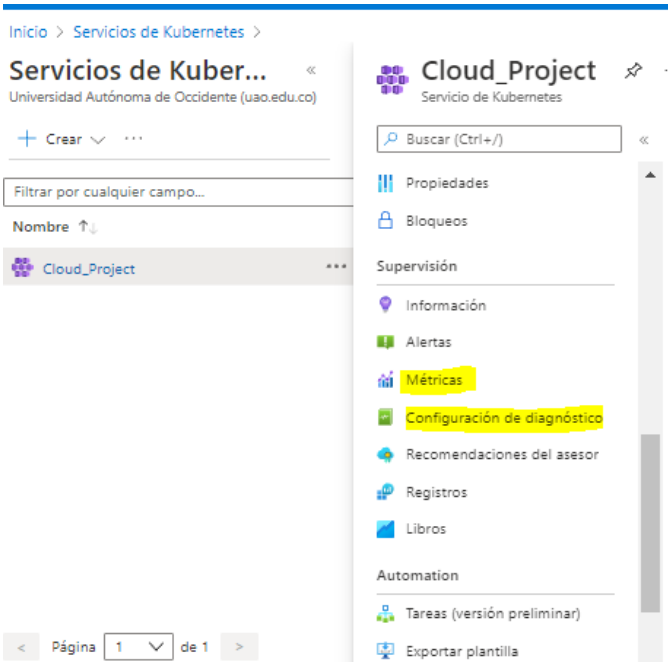
```
diana@Azure:~/nginx$ kubectl get pods -l app=nginx
NAME                                READY   STATUS              RESTARTS   AGE
nginx-deployment-559d658b74-cnpbt   0/1     ContainerCreating   0          2s
nginx-deployment-559d658b74-zdxlk   0/1     ContainerCreating   0          2s
diana@Azure:~/nginx$ kubectl get pods -l app=nginx
NAME                                READY   STATUS    RESTARTS   AGE
nginx-deployment-559d658b74-cnpbt   1/1     Running   0          8s
nginx-deployment-559d658b74-zdxlk   1/1     Running   0          8s
```

Verificamos el nuevo pod con nombre: nginx-deployment-559d658b74-cnpbt

```
Name:         nginx-deployment-559d658b74-cnpbt
Namespace:    default
Priority:     0
Node:         aks-agentpool-70570969-vmss000007/10.240.0.5
Start Time:   Fri, 12 Nov 2021 20:45:19 +0000
Labels:       app=nginx
              pod-template-hash=559d658b74
Annotations:  <none>
Status:       Running
IP:           10.244.0.14
IPs:
  IP:           10.244.0.14
Controlled By:  ReplicaSet/nginx-deployment-559d658b74
Containers:
  nginx:
    Container ID:   containerd://335a85d0b886666adf8be26ded03424a0de5b4c7e3dd309133a4d40f8b56d66e
    Image:          nginx:1.16.1
    Image ID:       docker.io/library/nginx@sha256:d20aa6d1cae56fd17cd458f4807e0de462caf2336f0b70b5eeb69fcaaf30dd9c
    Port:           80/TCP
    Host Port:      0/TCP
    State:          Running
      Started:      Fri, 12 Nov 2021 20:45:24 +0000
    Ready:          True
    Restart Count:  0
    Environment:    <none>
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from default-token-cx7tb (ro)
Conditions:
  Type              Status
  Initialized       True
  Ready             True
  ContainersReady   True
  PodScheduled      True
Volumes:
  default-token-cx7tb:
    Type:        Secret (a volume populated by a Secret)
    SecretName:  default-token-cx7tb
    Optional:    false
QoS Class:       BestEffort
Node-Selectors:  <none>
Tolerations:     node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
                 node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type    Reason     Age   From               Message
  ----    ------     ----  ----               -------
  Normal  Scheduled  33s   default-scheduler  Successfully assigned default/nginx-deployment-559d658b74-cnpbt to aks-agentpool-70570969-vmss000007
  Normal  Pulling    33s   kubelet            Pulling image "nginx:1.16.1"
  Normal  Pulled     29s   kubelet            Successfully pulled image "nginx:1.16.1" in 3.889125742s
  Normal  Created    28s   kubelet            Created container nginx
  Normal  Started    28s   kubelet            Started container nginx
```

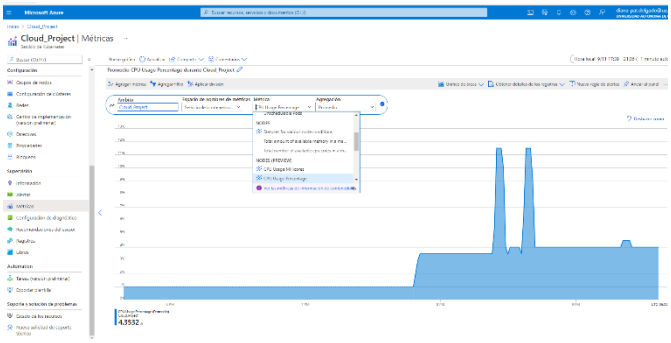
VI. REVISION DE METRICAS (SUPERVISION Y MONITOREO)

En la pantalla principal donde se ven las características del clúster, se puede verificar en el menú la opción de métricas y configuración de diagnóstico, la cual nos permite configurar la visualización del estado de nuestro clúster y los recursos que usa.

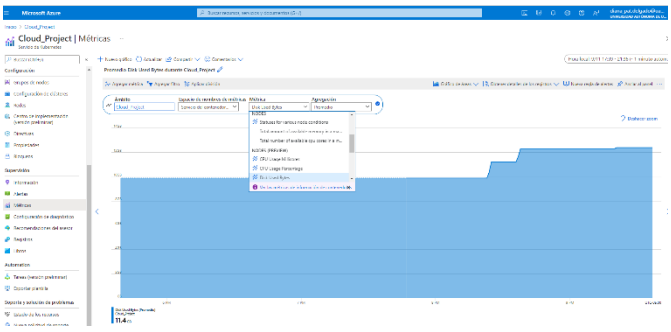


Desde estas opciones se pueden verificar los diferentes consumos y configurar el monitoreo de los recursos como se ve en las siguientes graficas:

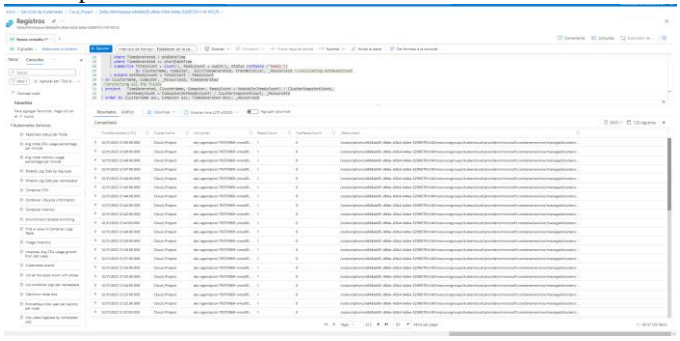
Desde las métricas se valida uso de la CPU:



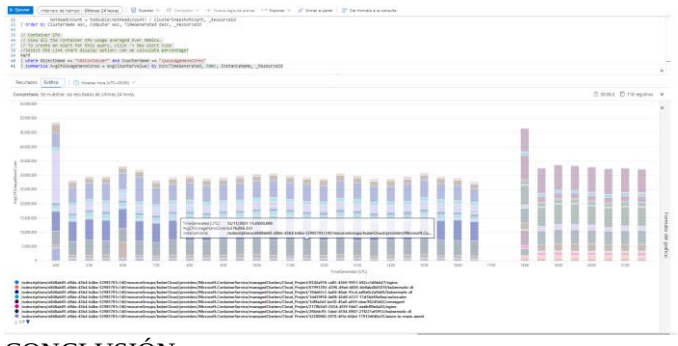
Uso del disco duro:



Desde la configuración de diagnóstico se evalúan las mismas opciones pero en un monitoreo continuo de los nodos:



Verificamos el uso de CPU de los nodos:



CONCLUSIÓN

- Desde Azure se tiene un manejo integrado de los recursos de un cluster de una manera mas rápida y se puede monitorear su funcionamiento en tiempo real.
- La creación del clúster y la validación del funcionamiento del mismo a través de la interfaz grafica de azure permite tener acceso a metrica y monitoreo de los recursos que usa el servicio implementado.
- mediante la funcionalidad de preseed(nginx) permite visualizar el escalamiento de los pods para laa tencion a microservicios mas facil.

References:

- PAGINA DE AYUDA DESPLEGAR APLICACION DE CLASIFICACION DE IMAGENES: https://opensource.com/article/20/9/deep-learning-model-kubernetes
- Conceptos básicos de Kubernetes de Azure Kubernetes Service (AKS): https://docs.microsoft.com/es-es/azure/aks/concepts-clusters-workloads
- Corre una aplicación stateless usando un Deployment: https://kubernetes.io/es/docs/tasks/run-application/run-stateless-application-deployment/