

Actividad 8 - Kubernetes



Estudiantes:

- Diaz Francisco
- Gonzalez Lautaro
- Peirone Santiago
- Zarza Valentín

Desplegar un clúster

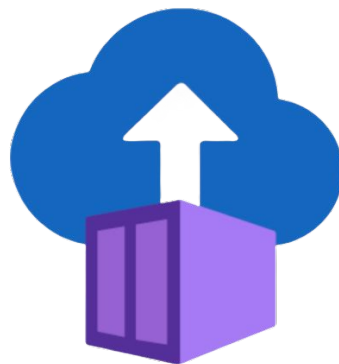
¿Que es un cluster?

El término clúster (del inglés *cluster*, que significa 'grupo' o 'racimo') se aplica a los **sistemas distribuidos** de **computadoras** unidos entre sí normalmente por una **red** de alta velocidad y que se comportan como si fuesen un único servidor.



¿Como lo despliego?

En nuestro caso utilizamos **Microsoft Azure** que cuenta con su propio servicio de kubernetes





¿Como lo despliego?

1

Para desplegarlo/crearlo nos tenemos que dirigir a la parte de servicios que nos brinda Azure.

Servicios de Azure

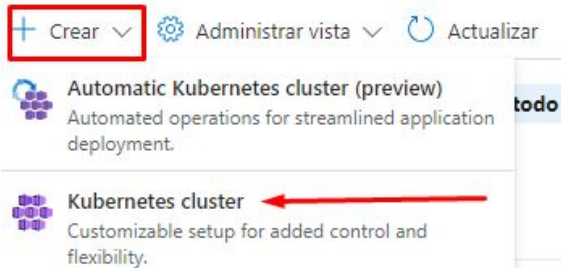


Crear un recurso



Servicios de Kubernetes

Una vez dentro nos dirigimos a “Crear” y seleccionamos “Kubernetes Cluster”



A continuación de eso Azure tiene una interfaz hecha para la creación y edición de dicho cluster, en esta podremos ponerle nombre, crear grupo de nodos, etc

Datos básicos

Región *

(South America) Brazil South

Zonas de disponibilidad

Ninguno

Plan de tarifa de AKS

Gratis

Versión de Kubernetes *

1.28.9 (predeterminado)

Grupos de nodos

Actualización automática

Programador de actualización automática

Habilitado con revisión (recomendado)

Cada semana el domingo (recomendado)

Inicio el: Sat Jun 22 2024 00:00 +00:00 (Hora universal coordinada)

[Editar programación](#)

Además del grupo de nodos, puedes crear grupos de nodos opcionales para controlar la configuración de red.

+ Agregar grupo de nodos

Eliminar



Nombre

Redes de contenedores

Configuración de red



agentpool

☐ Kubenet

Ideal para grupos de nodos más pequeños. A cada pod se le asigna una dirección IP lógicamente diferente de la subred para simplificar la configuración.

☒ Azure CNI

Ideal para grupos de nodos más grandes. A cada nodo y pod se le asigna una dirección IP única para las configuraciones avanzadas.

☐ Los valores altos de pod pueden agotar rápidamente las direcciones IP disponibles. [Más información](#)



¿Como lo despliego?

1

Lo **importante** acá es crear un grupo de “Nodos”(En el cual seleccionaremos un rango de cantidad de nodos)

Grupos de nodos

Además del grupo de nodos principal necesario configurado en la pestaña Conceptos básicos, también puede agregar grupos de nodos opcionales para controlar una gran variedad de cargas de trabajo. [Más información](#)

Un grupo de recursos para ese cluster (Este será nuestro “Servidor”)

Detalles del proyecto

Seleccione una suscripción para administrar los recursos implementados y los costos. Use los grupos de recursos como carpetas para organizar y administrar todos los recursos.

Suscripción * ①

Azure for Students

Grupo de recursos * ①

cant_group

[Crear nuevo](#)

Y por último configuramos el apartado de “Red” y de “Supervisión” para luego

Información de contenedores

Habilitar registros de contenedor



Se recomienda la Azure Monitor para la configuración de Desarrollo/pruebas.

Área de trabajo de Log Analytics * ①

DefaultWorkspace-2611a0c3-08b8-4685-bdc1-3dc9576011bc-CQ

[Crear nueva](#)

Load Balancer

¿Que es?

Es una herramienta que **distribuye** el **tráfico** de red entrante a través de varios **servidores** o **pods**, asegurando que ninguna máquina se sobrecargue y que los recursos se utilicen de manera eficiente.



Réplica de Wordpress

Para crear las réplicas de Wordpress en nuestro clúster vamos a necesitar de 2 recursos asociado a nuestros pods

Recursos Necesarios:

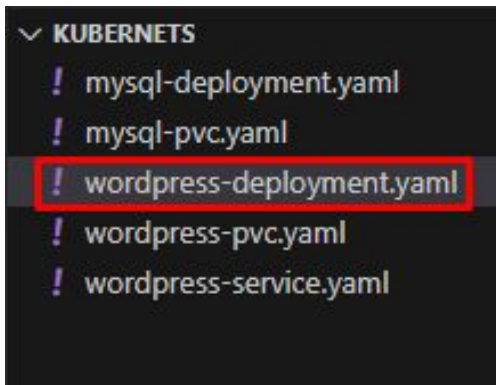
- **Deployment:** Un Deployment en Kubernetes se usa para desplegar y gestionar un conjunto de pods idénticos.
- **Service:** Un Service de tipo **LoadBalancer** se usa para distribuir el tráfico entre los pods del Deployment de WordPress.



Recursos

2

Para la creación del primer recurso vamos a necesitar de un editor de código. En este caso utilizaremos Visual Studio Code, para compilar dicho código.



Y desarrollamos el deploy, en este caso utilizamos uno prehecho

Creamos un archivo .yaml en el cual será nuestro deployment

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: wordpress
5  spec:
6    replicas: 2
7    selector:
8      matchLabels:
9        app: wordpress
10   template:
11     metadata:
12       labels:
13         app: wordpress
14     spec:
15       containers:
16         - name: wordpress
17           image: wordpress:latest
18           ports:
19             - containerPort: 80
20           env:
21             - name: WORDPRESS_DB_HOST
22               value: mysql-service.mysql.svc.cluster.local
23             - name: WORDPRESS_DB_NAME
24               value: wordpress
25             - name: WORDPRESS_DB_USER
26               value: admin
27             - name: WORDPRESS_DB_PASSWORD
28               valueFrom:
29                 secretKeyRef:
30                   name: mysql-secret
31                   key: password
32           volumeMounts:
33             - name: wordpress-persistent-storage
34               mountPath: /var/www/html
35       volumes:
36         - name: wordpress-persistent-storage
37           persistentVolumeClaim:
38             claimName: wordpress-pv-claim
39   ---
40   apiVersion: v1
41   kind: Service
42   metadata:
43     name: wordpress
44   spec:
45     type: LoadBalancer
46     ports:
47       - port: 80
48         targetPort: 80
49     selector:
50       app: wordpress
51
```



Recursos

2

Luego en el mismo editor creamos otro recurso, en este caso será nuestro “service” para la función posterior

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: wordpress
5  spec:
6    type: LoadBalancer
7    ports:
8      - port: 80
9        targetPort: 80
10   selector:
11     app: wordpress
```

Importante en las especificaciones agregarle el tipo

Una vez creado el “deploy” y el “service” nos falta un último recurso donde se almacenará de forma constante. El volumen persistente (PVC)

```
! wordpress-pvc.yaml
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: wordpress-pv-claim
5  spec:
6    accessModes:
7      - ReadWriteOnce
8    resources:
9      requests:
10        storage: 1Gi
```



Conexión al cluster

2

Para finalizar el load balancer con ambas réplicas de Wordpress, primero debemos conectarnos a el clúster o utilizar la terminal que nos ofrece Azure en su página. En nuestro caso lo que hicimos fue conectarnos a el clúster de forma remota.



Essentials	
Grupo de recursos	: grupoderecursos
Estado	: Succeeded (Running)
Suscripción	: Azure for Students
Ubicación	: Brazil South

Y en tu maquina hacer una conexión con los siguientes comandos:

Conectarse a prueba-cluster

Cloud Shell

CLI de Azure

Comando de ejecución

Conéctese al clúster mediante las herramientas de línea de comandos para interactuar directamente con el clúster mediante kubectl, la herramienta de línea de comandos para Kubernetes. Kubectl está disponible en Azure Cloud Shell de forma predeterminada y también se puede instalar localmente.

Requisitos previos

- 1 [Instalar la CLI de Azure](#)
- 2 [Instalación de kubectl](#)

La misma plataforma te guía, en nuestro caso lo hicimos a través de PowerShell

Establecer el contexto de clúster.

- 1 Abrir terminal
- 2 Ejecutar los siguientes comandos

Inicia sesión en la cuenta de Azure

```
az login
```

Establecer la suscripción del clúster

```
az account set --subscription 2611a0c3-08b8-4685-bdc1-3dc9576011bc
```

Descargar credenciales de clúster

```
az aks get-credentials --resource-group grupoderecursos --name prueba-cluste...
```




Conexión al cluster

2

```
Seleccionar Windows PowerShell
PS C:\Users\Santi> az login
Please select the account you want to log in with.

Retrieving tenants and subscriptions for the selection...

[Tenant and subscription selection]

No      Subscription name      Subscription ID
-----
[1] *   Azure for Students  2611a0c3-08b8-4685-bdc1-3dc9576

The default is marked with an *; the default tenant is 'Dir

Select a subscription and tenant (Type a number or Enter for

Tenant: Directorio predeterminado
Subscription: Azure for Students (2611a0c3-08b8-4685-bdc1-3

[Announcements]
With the new Azure CLI login experience, you can select the
36

If you encounter any problem, please open an issue at https

[Warning] The login output has been updated. Please be aware

PS C:\Users\Santi>
```

Una vez conectados solo nos resta aplicar los recursos creados previamente a el cluster.

Esto se hace con el comando:

kubectl apply -f (Seguido del nombre del archivo)

```
Seleccionar Windows PowerShell
PS C:\Users\Santi> kubectl get pods
NAME                                READY   STATUS    RESTARTS   AGE
mysql-deployment-8695577f84-8tsvw   0/1     Pending   0           86s
wordpress-5d6b967dc8-dfbqj         1/1     Running   0           21s
wordpress-5d6b967dc8-rkgk5         1/1     Running   0           9s
PS C:\Users\Santi>
```

Podemos utilizar el comando:

kubectl get pods

Para comprobar si se aplicó correctamente.

```
PS C:\Users\Santi> az account set --subscription 2611a0c3-08b8-4685-bdc1-3dc9576011bc
PS C:\Users\Santi> az aks get-credentials --resource-group grupoderecursos --name prueba-cluster --overwrite-existing
Merged "prueba-cluster" as current context in C:\Users\Santi\.kube\config
PS C:\Users\Santi>
```



Creación de volumen

3

Como explicamos anteriormente utilizamos otra vez el editor de código, en este desarrollamos el persistent volumen claim

```
! wordpress-pvc.yaml
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: wordpress-pv-claim
5  spec:
6    accessModes:
7      - ReadWriteOnce
8    resources:
9      requests:
10       storage: 1Gi
```

Y a continuación utilizamos el comando:

kubectl apply -f (Nombre del archivo que creamos en el editor de código)

```
PS C:\Users\Santi\Desktop\kubernets> kubectl apply -f wordpress-pvc.yaml
persistentvolumeclaim/wordpress-pv-claim unchanged
PS C:\Users\Santi\Desktop\kubernets> kubectl get pvc
NAME                                STATUS  VOLUME                                     CAPACITY
wordpress-pv-claim                 Bound   pvc-124addf5-522a-474d-936a-51cb29e58900  1Gi
PS C:\Users\Santi\Desktop\kubernets>
```



Creación de volumen

3

Y por otro lado desarrollamos el persistent volumen

```
! wordpress-pv.yaml
1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: wordpress-pvc-claim
5  spec:
6    capacity:
7      storage: 1Gi
8    accessModes:
9      - ReadWriteOnce
10   persistentVolumeReclaimPolicy: Retain
11   storageClassName: manual
12   hostPath:
13     path: "/mnt/data"
```

Y a continuación utilizamos el comando:

kubectl apply -f (Nombre del archivo que creamos en el editor de código)

```
PS C:\Users\Santi\Desktop\kubernets> kubectl apply -f wordpress-pv.yaml
persistentvolume/wordpress-pvc-claim created
```

<input type="checkbox"/>	Nombre	Espacio de nombres	Estado	Volumen	Capacidad	Modos de acc...
<input type="checkbox"/>	wordpress-pv-claim	default	✔ Bound	pvc-124addf5-522a-474...	1Gi	ReadWriteOnce



Deploy de MySQL

4

```
! mysql-deployment.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: mysql
5  spec:
6    selector:
7      matchLabels:
8        app: mysql
9    strategy:
10     type: Recreate
11    template:
12     metadata:
13       labels:
14         app: mysql
15     spec:
16       containers:
17         - image: mysql:5.7
18           name: mysql
19           env:
20             - name: admin
21               value: admin
22           ports:
23             - containerPort: 3306
24               name: mysql
25           volumeMounts:
26             - name: mysql-persistent-storage
27               mountPath: /var/lib/mysql
28       volumes:
29         - name: mysql-persistent-storage
30           persistentVolumeClaim:
31             claimName: mysql-pv-claim
32  ---
```

Creamos el deploy en nuestro editor de código

```
32  ---
33  apiVersion: v1
34  kind: Service
35  metadata:
36    name: mysql
37  spec:
38    ports:
39      - port: 3306
40    selector:
41      app: mysql
42  ---
```

Y lo aplicamos con el comando anteriormente usado

```
PS C:\Users\Santi\Desktop\kubernets> kubectl apply -f mysql-deployment.yaml
deployment.apps/mysql created
service/mysql created
```



Creación de volumen de MySQL

5

Para la creación del volumen volvemos a nuestro editor de código y creamos nuestros volúmenes PV y PVC

```
! mysql-pvc.yaml
1  apiVersion: v1
2  kind: PersistentVolumeClaim
3  metadata:
4    name: mysql-pv-claim
5  spec:
6    accessModes:
7      - ReadWriteOnce
8    resources:
9      requests:
10       storage: 1Gi
11     storageClassName: manual
12
```

```
! mysql-pv.yaml
1  apiVersion: v1
2  kind: PersistentVolume
3  metadata:
4    name: mysql-pv
5  spec:
6    capacity:
7      storage: 1Gi
8    accessModes:
9      - ReadWriteOnce
10   persistentVolumeReclaimPolicy: Retain
11   storageClassName: manual
12   hostPath:
13     path: "/mnt/mysql-data"
```

Y lo aplicamos utilizando el mismo comando usado anteriormente

```
PS C:\Users\Santi\Desktop\kubernets> kubectl apply -f mysql-pv.yaml
persistentvolume/mysql-pv created
PS C:\Users\Santi\Desktop\kubernets> kubectl apply -f mysql-pvc.yaml
persistentvolumeclaim/mysql-pv-claim created
PS C:\Users\Santi\Desktop\kubernets>
```



Herramienta de monitoreo

6

Azure ya cuenta con una herramienta de monitoreo, que es la que marcamos anteriormente cuando creamos el cluster. Esta herramienta de “supervisión” cuenta con varios detalles importantes

Información general

Registro de actividad

Control de acceso (IAM)

Etiquetas

Diagnosticar y solucionar problemas

Microsoft Defender for Cloud

> Recursos de Kubernetes

> Configuración

> Supervisión

> Automation

> Ayuda

Supervisión

Información

Alertas

Métricas

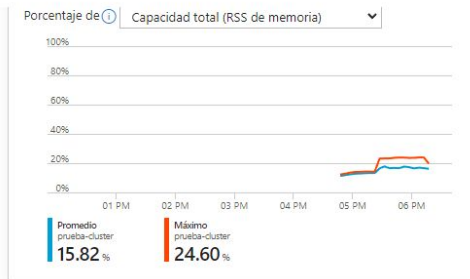
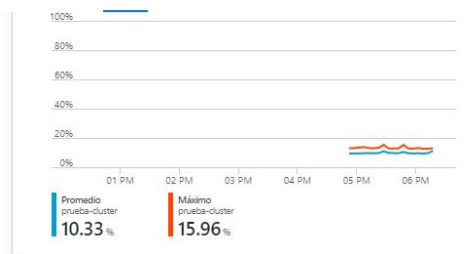
Configuración de diagnóstico

Recomendaciones del asesor

Registros

Libros

Novedades Clúster Informes Nodos Controladores Contenedores



Sino también existe un plugin en Azure llamado “Prometheus”



¿Que es un ingress?

Ingress es un **recurso** en Kubernetes que gestiona el acceso externo a los servicios del clúster, típicamente HTTP. Permite definir reglas de enrutamiento para manejar solicitudes basadas en rutas específicas (paths) o subdominios.

¿Que es un path?

Path (ruta) en el contexto de redes y servicios web, es una parte de una URL que especifica una ubicación en el servidor. Los paths permiten definir rutas específicas dentro de una aplicación web o servicio, que pueden dirigir a diferentes recursos o servicios.

¿Cómo genero un certificado gratuito?

Un certificado SSL/TLS gratuito puede ser generado utilizando **Let's Encrypt**, una autoridad certificadora que proporciona certificados gratuitos y automatizados para asegurar sitios web mediante HTTPS.



¿Como consigo mi dominio?

Por lo general Wordpress de por si te da un dominio para usar de forma “gratuita” pero si se necesita ser mas especifico hay varias opciones mejores para conseguir tu propio dominio

Algunos ejemplos:

- GoDaddy
- Namecheap
- Google Domains

