Tutorial azure para crear contenedores en kubernetes con docker

Juan Casado Ballesteros

Descargar los recursos del tutorial

El contendido del tutorial se encuentra en

```
git clone https://github.com/Azure-Samples/azure-voting-app-redis.git
```

Inciar localmente mediante docker-compose

Utilizaremos la herramienta docker-compose para crear una instancia local del contido descargado. La información sobre el contenido que vamos a crear se encuentra en el archivo **docker-compose.yaml**. Previamente habremos debido de haber iniciado el daemon de docker.

```
docker-compose up -d
```

Se descargarán los contenedores de: 'azure_example_tutorial_default' y 'tiangolo/uwsgi-nginx-flask:python3.6'. Desdede pip se intalará redis.

Comprobamos que se haya creado todo correctamente.

```
docker images
docker ps
```

Podremos acceder a la instancia local desde *http://localhost:8080*.

Finalizar la instancia local

Una vez hayamos comprobado que todo funciona correctamente podremos finalizar la instancia local mediante.

```
docker-compose down
```

Subir la imagen a azure

Para poderlo hacer deberemos tener instalado azure *az*, podremoc omprobarlo mediante

```
az --version
```

Ubicarnos en la subscipción correcta

En caso de que utilicemos distintas cuentas de azure o que dentro de la misma tengamos varias formas de pago distintas podremos cambiar de subscripción lkistando primero aquella subscripciones a las que tenemos acceso y posteriormente cambiando a la que deseemos usar.

```
az account list --output table
az account set --subscription "Azure for Students"
```

Crear un container registry

- Crear recursos de grupo
- Crear una instancia de Azure Container Registry
- Entrar en en el Azure Container Registry

```
az group create --name ${resourceGroup} --location eastus
az acr create --resource-group ${resourceGroup} \
--name ${resource} --sku Basic
az acr login --name ${resource}
```

Preparar la imagen y subirla

Debemoc proporcionar un tag apropiado a las imágenes que queramos subir pues el tag es usado para redireccionarlas correctamente a la ubicación que deseemos. Debemos obtener el loginServer lo cual lo podemos hacer mediante:

```
az acr list --resource-group ${resourceGroup} \
--query "[].{acrLoginServer:loginServer}" --output table
```

Utilizamos ahora el login server para poporcionar el tag corrspopndiente a la imagen que creamos previamente.

```
docker-compose build
docker tag azure-vote-front ${resourceLogin}/azure-vote-front:v1
```

Una vez la imagen tenga el tag podremos subir la a azure:

```
docker push ${resourceLogin}/azure-vote-front:v1
```

Podremos comprobar que la imagen ha sido correctamente subida a Auzere. Para ello debemos preguntar al container registry (no al grupo). La hemos subido como v1 lo cual iondica la versión de la imagen.

```
az acr repository list --name ${resource} --output table
az acr repository show-tags --name ${resource} \
--repository azure-vote-front --output table
```

Crear un cluster de kubernetes

Utilizaremos el Azure Active Directory para indicar al cluster de kubernetes con qué recursos de azure puede interactuar. Crearemos un service principal tras lo que nos proporcionará una cuenta cuya información deberemos guardar. En concreto nos interesan los campos de *appID* y *password*

```
az ad sp create-for-rbac --skip-assignment
```

Proporcionar permiso al cluster

Primero obtendremos la ID de nuestro recurso y posteriormente la utilizamos para dar permisos sobre él a la aplicación.

```
az acr show --resource-group ${resourceGroup} \
--name ${resource} --query "id" --output tsv
az role assignment create --assignee ${appId} \
--scope ${resourceId} --role acrpull
```

Crear el cluster

Crearemos un cluster de kubernetes y nos conectaremos a él

```
az aks create \
--resource-group ${resourceGroup} \
--name ${clusterName} \
--node-count 1 \
--service-principal ${appId} \
--client-secret ${password} \
--generate-ssh-keys
az aks install-cli
az aks get-credentials --resource-group ${resourceGroup} \
--name ${clusterName}
kubectl get nodes
```

Con *kubectl get nodes* podremos comprobar que el proceso se haya realizado correctamente

Iniciar la aplicación

Lanzaremos públicamente la aplicación para que se pueda acceder a ella

Actualizar el manifest de kubernetes

Debemos incluir el recurso en el manifiesto de kubernetes para lo que utilizaremos el resourceLogin el cual deberemos de incluir en **azure-vote-all-in-one-redis.yaml** en el apartado de image de modo que quede como:

```
image: ${resourceLogin}.azurecr.io/azure-vote-front:v1
```

Iniciar la aplicación

Iniciaremos la aplicación desde kubernetes. Esto hará pública nuestra aplicación.

```
kubectl apply -f azure-vote-all-in-one-redis.yaml
```

Probar la aplicación

Podemos monitorizar la aplicación, al lanzar el comando obtendremos también la IP desde la que podremos acceder e ella.

```
kubectl get service azure-vote-front --watch
```

También podremos ver el estado de los contenedores que hayamos lanzado con kubernetes.

```
kubectl get pods
```

Escalar la aplicación

Podremos configurar el cluster de modo que se adapte a los requerientos que lo susiuarios hagan de él.

Escalado manual

Podeomos crear nuevas réplicas de la aplicación haciendo que haya tantas como indiquemos (ya sean más que las que hay actualmente o menos)

```
kubectl scale --replicas=${REPLICAS} deployment/azure-vote-front
```

Escalado automático

Para iniciar el escalado automático debemos comprobar que la versión de nuestro cluster sea superior o igual a 1.10

```
az aks show --resource-group ${resourceGroup} \
--name ${clusterName} --query kubernetesVersion
```

El escalado se realizará según el achivo .yml que confugura el cluster con resperto a los parámetros:

```
resources:
requests:
  cpu: 250m
limits:
  cpu: 500m
```

En este caso se indica que el escalado comenzará cuando el uso de cpu se aproxime al 50%

Podremos configurar el rango de réplicas válido para que nuestra aplicación oscile en él.

```
kubectl autoscale deployment azure-vote-front \
--cpu-percent=50 --min=3 --max=10
```

Escalado manual de nodos

Se podrá cambiar manualmente el número de nodos del cluster mediante:

```
az aks scale --resource-group ${resourceGroup} \
--name ${clusterName} --node-count 3
```

Actualizar la aplicación

Repitiendo los pasos hasta ahora explicados podremos actualizar la aplicación. Una de las ventajas que resaltan a primera vista sobre el uso de azure con contenedores a la hora de desarrollar aplicaciones es que evitamos completamente los estado de inestabilidad de cara al usuario. La aplicación publicada funcionará siempre y solo la actualizaremos cuando tengamos una nueva versión de esta que sea completamente funcional. Mientras estemos creando la nueva aplicación podremos probarla localmente sin riesgo de afectar a que es pública.

Las ventajas respecto de la virtualización y el auto escalado de la aplicación son inmediatas al publicarla y de un coste muy bajo en tiempo especialmente al comparar con las grandes ventajas que nuestra aplicación adquiere de cara a los usuarios.

```
docker-compose up --build -d
docker tag azure-vote-front ${resourceLogin}/azure-vote-front:v2
az acr login --name ${resource}
docker push ${resourceLogin}/azure-vote-front:v2
#ACTUALIZAR EL MANIFEST CON EL NUEVO NOMBRE!!!
kubectl set image deployment \
azure-vote-front azure-vote-front=${resourceLogin}/azure-vote-front:v2
#SOLO SI QUEREMOS CAMBIAR LA CONFIGURACION
kubectl delete hpa azure-vote-front
kubectl autoscale deployment azure-vote-front \
--cpu-percent=50 --min=2 --max=5
az aks scale --resource-group ${resourceGroup} \
--name ${clusterName} --node-count 2
```

Actualizar el cluster

Podemos obtener las versiones disponibles del cluster, actualizarlo a una de las versiones disponibles y finalmente verficar el estado de la actualización.

```
az aks get-upgrades --resource-group ${resourceGroup} \
--name ${clusterName} --output table
az aks upgrade --resource-group ${resourceGroup} \
--name ${clusterName} --kubernetes-version 1.13.5
az aks show --resource-group ${resourceGroup} \
--name ${clusterName} --output table
```

Borrar los recursos

La forma de garantizar que todos los recursos que hemos usado son eliminados y que no se nos podrá cobrar más por ellos es borrar el grupo de recursos.

```
az group delete --name ${resourceGroup} --yes --no-wait
```