Revisión de Azure Kubernetes Service (AKS)

Dado que el desarrollo de aplicaciones evoluciona hacia un enfoque basado en contenidos, cobra importancia la necesidad de organizar y administrar los recursos. Kubernetes es la plataforma líder que ofrece la capacidad de proporcionar programación de confianza de cargas de trabajo de aplicación con tolerancia a errores. Azure Kubernetes Service (AKS) es un oferta de Kubernetes administrado que simplifica aún más la administración e implementación de aplicaciones basadas en contenedores.

Kubernetes es...

Kubernetes es una plataforma de rápida evolución que administra aplicaciones basadas en contenedores y sus componentes de red y almacenamiento asociados. El foco está en las cargas de trabajo de la aplicación, no en los componentes de infraestructura subyacente. Kubernetes proporciona un enfoque declarativo en las implementaciones, respaldado por un sólido conjunto de API para las operaciones de administración.

Puede compilar y ejecutar aplicaciones modernas, portátiles y basadas en microservicios que se benefician de Kubernetes mediante la orquestación y administración de la disponibilidad de esos componentes de la aplicación. Kubernetes admite tanto aplicaciones con estado como sin estado, a medida que los equipos progresan a través de la adopción de aplicaciones basadas en microservicios.

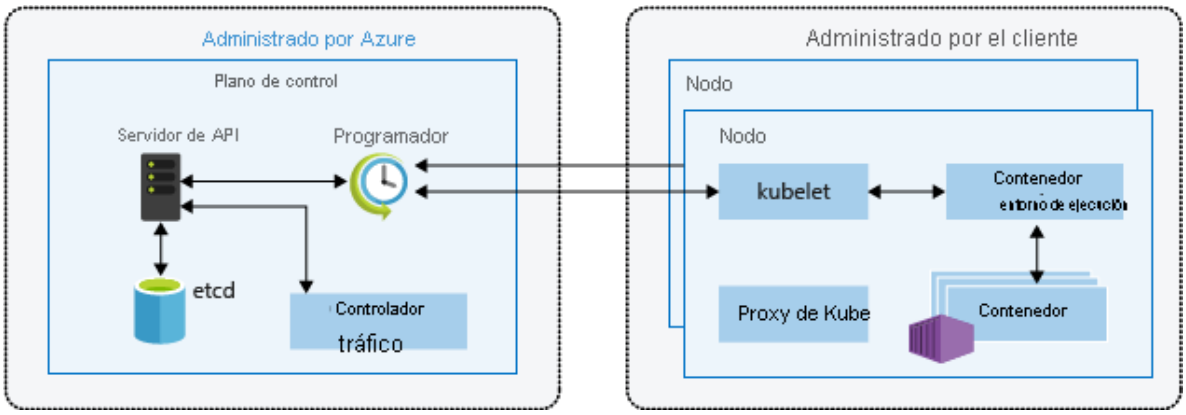
Como plataforma abierta, Kubernetes le permite compilar aplicaciones con el lenguaje de programación, el sistema operativo, las bibliotecas o el bus de mensajería que prefiera. La integración continua y las herramientas de entrega continua (CI/CD) existentes pueden integrarse con Kubernetes para programar e implementar versiones.

Azure Kubernetes Service (AKS) proporciona un servicio de Kubernetes administrado que reduce la complejidad de las principales tareas de administración e implementación, incluida la coordinación de actualizaciones. El plano de control de AKS es administrado por la plataforma de Azure, y solo paga por los nodos de AKS que ejecutan sus aplicaciones. AKS se ha diseñado sobre el motor de código abierto de Azure Kubernetes Service (aks-engine).

Arquitectura del clúster de Kubernetes

Un clúster de Kubernetes se divide en dos componentes:

- Los nodos del *plano de control* proporcionan los servicios centrales de Kubernetes y la orquestación de las cargas de trabajo de las aplicaciones.
- Los *nodos* ejecutan las cargas de trabajo de la aplicación.



Características de Azure Kubernetes Service

- Completamente administrada
- IP pública y FQDN (opción de IP privada)
- Acceso con RBAC o Azure AD
- Implementación de contenedores
- Contenedores de escalado dinámico
- Automatización de las actualizaciones graduales y de las reversiones de contenedores
- Administración del almacenamiento, el tráfico de red y la información confidencial

Implementación de una arquitectura de Azure Kubernetes Service

La arquitectura de clústeres de Kubernetes es un conjunto de recomendaciones de diseño para implementar sus contenedores en una configuración segura y administrada.

Patrón de clúster

Al crear un clúster de AKS, se crea y se configura automáticamente un patrón de clúster. Este patrón de clúster se proporciona como un recurso de Azure administrado que se extrae del usuario. No hay ningún costo para el patrón de clúster, solo los nodos que forman parte del clúster de AKS.

El patrón de clúster incluye los siguientes componentes principales de Kubernetes:

- **kube-apiserver**: el servidor de la API es el modo en el que se exponen las API de Kubernetes subyacentes. Este componente proporciona la interacción de las herramientas de administración, como kubectl o el panel de Kubernetes.

- etcd: para mantener el estado del clúster de Kubernetes y la configuración, el componente *etcd* de alta disponibilidad es un valor clave en Kubernetes.
- kube-scheduler: al crear o escalar aplicaciones, Scheduler determina qué nodos pueden ejecutar la carga de trabajo y los inicia.
- kube-controller-manager: el administrador de controladores supervisa un número de controladores más pequeños que realizan acciones como la replicación de los pods y el control de las operaciones del nodo.

AKS proporciona un patrón de clúster de un solo inquilino, con un servidor de API dedicado, programador, etc. Defina el número y el tamaño de los nodos, y la plataforma Azure configura la comunicación segura entre el patrón de clúster y los nodos. La interacción con el patrón de clúster se produce a través de las API de Kubernetes, como kubectl o el panel de Kubernetes.

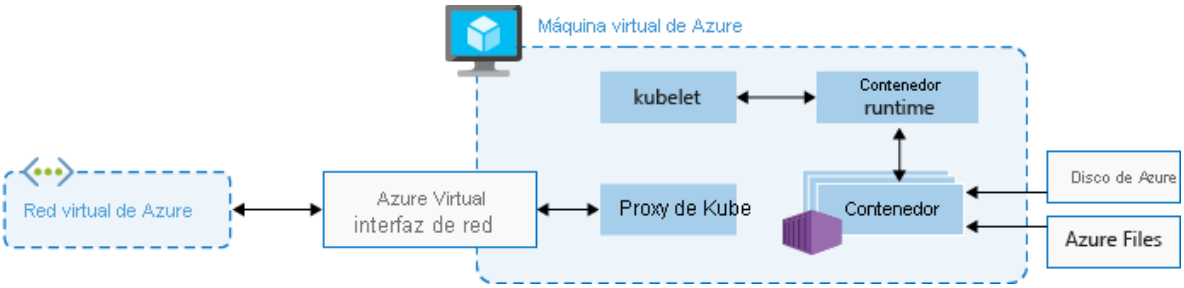
Este patrón de clúster administrado implica que no es necesario configurar componentes como un almacén de alta disponibilidad, pero también implica que no se puede acceder directamente al patrón del clúster. Las actualizaciones de Kubernetes se organizan a través de la CLI de Azure o Azure Portal, que actualiza el patrón de clúster y, a continuación, los nodos. Para solucionar posibles problemas, puede revisar los registros del patrón de clúster a través de Azure Log Analytics.

Si necesita configurar el patrón del clúster de una manera determinada o necesita acceso directo a este, puede implementar su propio clúster de Kubernetes mediante aks-engine.

Nodos y grupos de nodos

Para ejecutar las aplicaciones y los servicios de soporte técnico, necesitará un nodo de Kubernetes. Un clúster de AKS tiene uno o varios nodos, que consiste en una máquina virtual (VM) de Azure que ejecuta los componentes del nodo de Kubernetes y el entorno de ejecución del contenedor:

- El kubelet es el agente de Kubernetes que procesa las solicitudes de orquestación del plano de control y la programación de la ejecución de los contenedores solicitados.
- Las redes virtuales se controlan mediante kube-proxy en cada nodo. El proxy enruta el tráfico de red y administra las direcciones IP para los servicios y los pods.
- El *entorno de ejecución del contenedor* es el componente que permite que las aplicaciones en contenedor ejecuten recursos adicionales e interactúen con ellos, como la red virtual y el almacenamiento. En AKS, Moby se usa como el entorno de ejecución del contenedor.



El tamaño de la máquina virtual de Azure para los nodos define el número de CPU, la cantidad de memoria y el tamaño y tipo de almacenamiento disponible (por ejemplo, SSD de alto rendimiento o HDD normal). Si prevé que las aplicaciones requerirán gran cantidad de CPU y memoria o almacenamiento de alto rendimiento, planifique el tamaño del nodo en consecuencia. También puede escalar horizontalmente el número de nodos del clúster de AKS para satisfacer la demanda.

En AKS, la imagen de la máquina virtual para los nodos del clúster se basa en Ubuntu Linux o en Windows Server 2019. Al crear un clúster de AKS o escalar horizontalmente el número de nodos, la plataforma Azure crea el número solicitado de máquinas virtuales y las configura. No se puede realizar ninguna configuración manual. Los nodos de agente se facturan como máquinas virtuales estándar, por lo que cualquier descuento que tenga en el tamaño de la máquina virtual que esté utilizando (incluidas las reservas de Azure) se aplica automáticamente.

Si tiene que utilizar un sistema operativo de host diferente, otro entorno de ejecución del contenedor o incluir paquetes personalizados, puede implementar su propio clúster de Kubernetes mediante aks-engine. El motor aks-engine ascendente publica características y proporciona opciones de configuración antes de que se admitan oficialmente en los clústeres de AKS. Por ejemplo, si desea usar un entorno de ejecución de contenedor distinto de Moby, puede usar aks-engine para configurar e implementar un clúster de Kubernetes que satisfaga sus necesidades actuales.

Los nodos del patrón de clúster proporcionan los servicios principales de Kubernetes y la orquestación de cargas de trabajo de la aplicación. Los nodos (máquinas virtuales) ejecutan las cargas de trabajo de aplicación.

Terminología de AKS

Término

Descripción

Grupos

Grupo de nodos con una configuración idéntica

Node

Máquina virtual individual que ejecuta aplicaciones en contenedores

Pods

Instancia única de una aplicación. Un pod puede contener varios contenedores.

Implementación

Uno o varios pods idénticos administrados por Kubernetes.

de manifiesto

Archivo YAML que describe una implementación

Seguridad de componentes maestros

En AKS, los componentes maestros de Kubernetes forman parte del servicio administrado que proporciona Microsoft. Cada clúster de AKS tiene su propio maestro de Kubernetes dedicado con un solo inquilino para proporcionar el servidor de API, Scheduler, etc. Microsoft administra y mantiene este maestro.

De forma predeterminada, el servidor de API de Kubernetes utiliza una dirección IP pública y un nombre de dominio completo (FQDN). Puede controlar el acceso al servidor de API mediante controles de acceso basado en rol y Azure Active Directory.

Seguridad de nodos

Los nodos de AKS son máquinas virtuales de Azure de los que usted realiza la administración y el mantenimiento. Los nodos de Linux ejecutan una distribución Ubuntu optimizada con el tiempo de ejecución del contenedor de Moby. Los nodos de Windows Server ejecutan una versión de Windows Server 2019 y también, el entorno de ejecución del contenedor de Moby. Cuando se crea o se escala verticalmente un clúster de AKS, los nodos se implementan automáticamente con las actualizaciones de seguridad del sistema operativo y las configuraciones más recientes.

La plataforma Azure aplica automáticamente las revisiones de seguridad del sistema operativo a los nodos de Linux del clúster durante la noche. Si una actualización de seguridad del sistema operativo de Linux requiere un reinicio del host, este no se realiza automáticamente. Puede reiniciar manualmente los nodos de Linux o bien optar por un enfoque común que consiste en usar Kured, un demonio de reinicio de código abierto para Kubernetes. Kured se ejecuta como un elemento DaemonSet y supervisa cada nodo para comprobar la presencia de un archivo que indique que hace falta un reinicio. Los reinicios se administran en el clúster con el mismo proceso de acordonar y purgar que una actualización de clúster.

Para los nodos de Windows Server, Windows Update no ejecuta ni aplica las actualizaciones más recientes de manera automática. En una programación normal del

ciclo de versiones de Windows Update y su proceso de validación propio, debe realizar una actualización de los grupos de nodos de Windows Server en el clúster de AKS. Este proceso de actualización crea nodos que ejecutan la imagen y las revisiones más recientes de Windows Server y elimina los nodos anteriores. Los nodos se implementan en una subred de una red privada virtual, sin ninguna dirección IP pública asignada. Para fines de administración y solución de problemas, SSH está habilitado de forma predeterminada. El acceso de SSH solo está disponible mediante la dirección IP interna.

Para proporcionar almacenamiento, los nodos usan Azure Managed Disks. Para la mayoría de los tamaños de nodo de máquina virtual, estos son los discos Premium respaldados por SSD de alto rendimiento. Los datos almacenados en discos administrados se cifran automáticamente en reposo dentro de la plataforma Azure. Para mejorar la redundancia, estos discos también se replican de forma segura en el centro de datos de Azure.

Los entornos de Kubernetes en AKS o en cualquier otro lugar no están completamente seguros en este momento ante la utilización de multiinquilinos hostiles. Utilizar otras características de seguridad adicionales, como las directivas de seguridad de pod o los controles de acceso basados en roles (RBAC) más específicos, puede dificultar las vulnerabilidades de seguridad. Sin embargo, para que la seguridad resulte efectiva cuando se ejecutan cargas de trabajo multiinquilino hostiles, el hipervisor es el único nivel de seguridad en el que debe confiar. El dominio de seguridad de Kubernetes se convierte en todo el clúster, no en un nodo específico. En el caso de estos tipos de cargas de trabajo multiinquilino hostiles, debe usar clústeres que estén físicamente aislados.

Para proteger los datos de los clientes a medida que ejecuta cargas de trabajo de aplicaciones en Azure Kubernetes Service (AKS), es importante tener en cuenta la seguridad del clúster.

Configuración de redes de Azure Kubernetes Service

Para permitir el acceso a las aplicaciones o para que los componentes de las aplicaciones se comuniquen entre sí, Kubernetes proporciona una capa de abstracción para la red virtual. Los nodos de Kubernetes están conectados a una red virtual y pueden proporcionar conectividad de entrada y salida para los pods. El componente kube-proxy se ejecuta en cada nodo para proporcionar estas características de red.

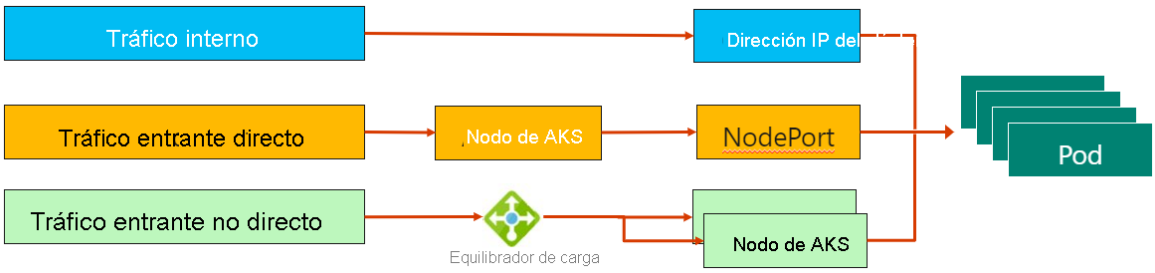
En Kubernetes, los servicios agrupan lógicamente los pods para permitir el acceso directo mediante una dirección IP o nombre DNS y en un puerto específico. También puede distribuir el tráfico mediante un equilibrador de carga. También se puede lograr un enrutamiento del tráfico de la aplicación más complejo mediante los controladores de entradas. La seguridad y el filtrado del tráfico de red de los pods es posible con las directivas de red de Kubernetes.

La plataforma Azure también ayuda a simplificar las redes virtuales de los clústeres de AKS. Cuando se crea un equilibrador de carga de Kubernetes, se crea y se configura el recurso de equilibrador de carga de Azure subyacente. Cuando abre los puertos de red a los pods, se configuran las reglas de los grupos de seguridad de red de Azure correspondientes. Para el enrutamiento de aplicaciones HTTP, Azure también puede configurar un DNS externo a medida que se configuran nuevas rutas de entrada.

Servicios

Para simplificar la configuración de red de las cargas de trabajo de la aplicación, Kubernetes usa servicios para agrupar lógicamente un conjunto de pods y proporcionar conectividad de red. Están disponibles los siguientes tipos de servicio:

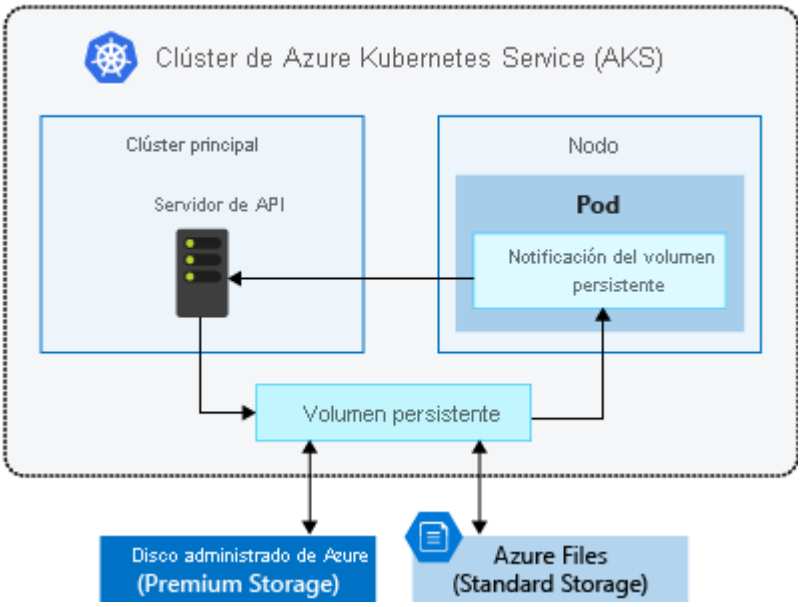
- **Dirección IP de clúster**: crea una dirección IP interna para su uso dentro del clúster de AKS. Es útil solo para aplicaciones internas que admiten otras cargas de trabajo dentro del clúster.
- **NodePort**: crea una asignación de puerto en el nodo subyacente que permite que se pueda acceder a la aplicación directamente con la dirección IP del nodo y el puerto.
- **LoadBalancer**: crea un recurso de equilibrador de carga de Azure, configura una dirección IP externa y conecta los pods solicitados al grupo de back-end del equilibrador de carga. Para permitir que el tráfico de los clientes llegue a la aplicación, se crean reglas de equilibrio de carga en los puertos deseados.
- **ExternalName**: crea una entrada DNS específica para facilitar el acceso a la aplicación.



Al ejecutar aplicaciones modernas basadas en microservicios en Kubernetes, con frecuenta querrá controlar qué componentes pueden comunicarse entre sí. El **principio de privilegios mínimos** debe aplicarse a cómo puede fluir el tráfico entre pods en un clúster Azure Kubernetes Service (AKS). Supongamos que quiere bloquear el tráfico directamente a las aplicaciones de back-end. La característica *Directiva de red* de Kubernetes permite definir las reglas de tráfico de entrada y salida entre pods de un clúster.

Implementación del almacenamiento de Azure Container Service

Las aplicaciones que se ejecutan en Azure Kubernetes Service (AKS) pueden necesitar almacenar y recuperar datos. Para algunas cargas de trabajo de la aplicación, este almacenamiento de datos puede usar el almacenamiento rápido local en el nodo que ya no es necesario cuando se eliminan los pods. Otras cargas de trabajo de la aplicación pueden requerir almacenamiento que conserve en volúmenes de datos más regulares dentro de la plataforma Azure. Es posible varios pods necesiten compartir los mismos volúmenes de datos o volver a conectar los volúmenes de datos si se vuelve a programar el pod en otro nodo. Por último, es posible que deba insertar datos confidenciales o información de configuración de la aplicación en los pods.



Volúmenes

A menudo, las aplicaciones necesitan poder almacenar y recuperar datos. Dado que Kubernetes suele tratar los pods individuales como recursos efímeros y descartables, existen diferentes enfoques disponibles para usar aplicaciones y conservar datos según sea necesario. **Un volumen representa una forma de almacenar, recuperar y conservar datos entre pods y a través del ciclo de vida de la aplicación**.

Los volúmenes tradicionales para almacenar y recuperar datos se crean como recursos de Kubernetes respaldados por Azure Storage. Puede crear manualmente estos volúmenes de datos para que se asignen directamente a los pods, o hacer que Kubernetes los cree automáticamente. Estos volúmenes de datos pueden utilizar Azure Disks o Azure Files:

- **Azure Disks** puede usarse para crear un recurso DataDisk de Kubernetes. Azure Disks puede usar almacenamiento Premium de Azure, respaldado por SSD de alto rendimiento, o bien almacenamiento estándar de Azure, respaldado por unidades

de disco duro normales. Para la mayoría de las cargas de trabajo de producción y desarrollo, utilice el almacenamiento Premium. Los discos de Azure Disks se montan como ReadWriteOnce, por lo que solo están disponibles para un único pod. Para los volúmenes de almacenamiento a los que pueden acceder varios pods simultáneamente, use Azure Files.

- Azure Files se puede usar para montar un recurso compartido de SMB 3.0 respaldado por una cuenta de Azure Storage en los pods. Azure Files permite compartir datos entre varios nodos y pods. Los archivos pueden usar almacenamiento Standard de Azure, respaldado por HDD normales o almacenamiento Premium de Azure respaldado por SSD de alto rendimiento.

Volúmenes persistentes

Los volúmenes que se definen y se crean como parte del ciclo de vida del pod solo existen hasta que se elimina este. Los pods suelen esperar que su almacenamiento se conserve si un pod se vuelve a programar en un host diferente durante un evento de mantenimiento, especialmente en StatefulSets. Un volumen persistente es un recurso de almacenamiento creado y administrado por la API de Kubernetes, que puede existir más allá de la duración de un pod individual.

Se usan Azure Disks o Azure Files para proporcionar el volumen PersistentVolume. Como se indicó en la sección anterior sobre los volúmenes, la elección de Azure Disks o Azure Files suele venir determinada por la necesidad de acceso simultáneo a los datos o al nivel de rendimiento.

Un administrador de clústeres puede crear *estáticamente* un PersistentVolume o el servidor de API de Kubernetes puede crearlo *dinámicamente*. Si un pod está programado y solicita almacenamiento que no está disponible actualmente, Kubernetes puede crear el almacenamiento subyacente de Azure Disks o Azure Files y conectarlo al pod. El aprovisionamiento dinámico usa una clase StorageClass para identificar qué tipo de almacenamiento de Azure debe crearse.

Clases de almacenamiento

Para definir niveles de almacenamiento diferentes, como Premium y Estándar, puede crear una clase StorageClass. La clase StorageClass también define la directiva reclaimPolicy. Esta directiva reclaimPolicy controla el comportamiento del recurso de almacenamiento subyacente de Azure cuando se elimina el pod y el volumen persistente puede dejar de ser necesario. El recurso de almacenamiento subyacente se puede eliminar o conservar para su uso con un pod futuro.

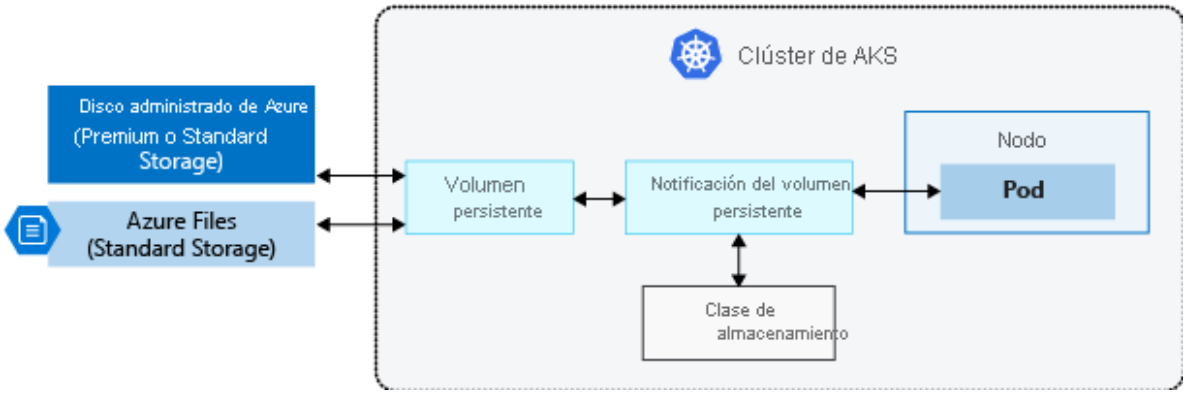
En AKS, se crean dos clases StorageClass iniciales:

- **default**: usa el almacenamiento estándar de Azure para crear un disco administrado. La directiva de reclamación indica que el almacenamiento de Azure Disks subyacente se elimina cuando se elimina el volumen persistente que lo utiliza.
- **managed-premium**: utiliza el almacenamiento Premium de Azure para crear un disco administrado. De nuevo, la directiva de reclamación indica que el almacenamiento de Azure Disks subyacente se elimina cuando se elimina el volumen persistente que lo utiliza.

Si no se especifica ninguna clase StorageClass para un volumen persistente, se usa el valor de StorageClass predeterminado.

Notificaciones de volúmenes persistentes

Una notificación PersistentVolumeClaim solicita almacenamiento en disco o archivo de una clase StorageClass, un modo de acceso y un tamaño determinados. El servidor de API de Kubernetes puede aprovisionar dinámicamente el recurso de almacenamiento subyacente de Azure si no hay ningún recurso existente para satisfacer la notificación de acuerdo con la clase StorageClass definida. La definición del pod incluye el montaje del volumen una vez que se este se ha conectado al pod.



Un volumen PersistentVolume se enlaza a una notificación PersistentVolumeClaim cuando se asigna un recurso de almacenamiento disponible al pod que lo solicita. Existe una asignación de 1:1 de volúmenes a notificaciones.

Protección de la autenticación en Azure Kubernetes Service con Active Directory

Hay diferentes maneras de autenticarse y proteger los clústeres de Kubernetes. Con el control de acceso basado en rol (RBAC), puede conceder a los usuarios o grupos acceso solo a los recursos que necesitan. Con Azure Kubernetes Service (AKS), puede mejorar aún más la seguridad y la estructura de permisos mediante el uso de Azure Active Directory. Estos enfoques le ayudan a proteger las cargas de trabajo de sus aplicaciones y los datos del cliente.

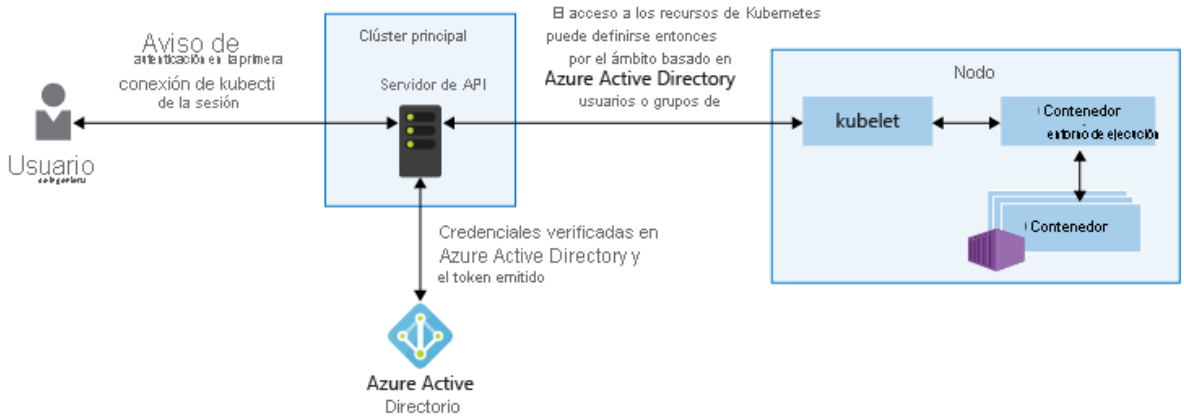
Cuentas de servicio de Kubernetes

Uno de los tipos de usuario principales de Kubernetes es una cuenta de servicio. Una cuenta de servicio existe en la API de Kubernetes y está administrada por esta. Las credenciales de las cuentas de servicio se almacenan como secretos de Kubernetes que usan los pods autorizados para comunicarse con el servidor de API. La mayoría de las solicitudes de API proporcionan un token de autenticación para una cuenta de servicio o una cuenta de usuario normal.

Las cuentas de usuario normales permiten un acceso más tradicional para administradores o desarrolladores humanos, no solo a servicios y procesos. Kubernetes por sí solo no proporciona una solución de administración de identidades donde se almacenan contraseñas y cuentas de usuario normales. En su lugar, se pueden integrar soluciones de identidad externas en Kubernetes. Para los clústeres de AKS, esta solución de identidades integrada es Azure Active Directory.

Integración de Azure Active Directory

Puede mejorar la seguridad de los clústeres de AKS con la integración de Azure Active Directory (AD). Compilada después de varias décadas de administración de identidades empresariales, Azure AD es un servicio multiinquilino de administración de identidades y de directorios basado en la nube que combina los servicios de directorio principales, la administración del acceso de las aplicaciones y la protección de identidades. Con Azure AD, puede integrar identidades locales en clústeres de AKS para proporcionar un único origen para la seguridad y administración de cuentas.



Con los clústeres de AKS integrados en Azure AD, puede conceder a los usuarios o grupos acceso a los recursos de Kubernetes de un espacio de nombres o del clúster. Para obtener un contexto de configuración de kubectl, un usuario puede ejecutar el comando az aks get-credentials. Cuando un usuario interactúa con el clúster de AKS con kubectl, se le pide que inicie sesión con sus credenciales de Azure AD. Este enfoque proporciona un único origen para la administración de cuentas de usuario y de las credenciales de contraseña. El usuario solo puede acceder a los recursos como defina el administrador de clústeres.

La autenticación de Azure AD en los clústeres de AKS usa OpenID Connect, una capa de identidad que se basa en el protocolo OAuth 2.0. OAuth 2.0 define los mecanismos para obtener y utilizar tokens de acceso para acceder a recursos protegidos, y OpenID Connect implementa la autenticación como una extensión del proceso de autorización de OAuth 2.0.

Administración del acceso a Azure Kubernetes Service mediante controles de acceso basado en rol de Azure

Un mecanismo adicional para controlar el acceso a los recursos es el control de acceso de basado en rol (RBAC) de Azure. El control de acceso basado en rol de Kubernetes está diseñado para trabajar con los recursos del clúster de AKS y el de Azure está diseñado para trabajar con los recursos de su suscripción de Azure. Con el control de acceso basado en rol de Azure, puede crear una definición de rol que describe los permisos que se aplicarán. A continuación, se asigna esta definición de rol a un usuario o grupo para un ámbito determinado que puede ser un recurso individual, un grupo de recursos o en toda la suscripción.

Roles y ClusterRoles

Antes de asignar permisos a los usuarios con el control de acceso basado en rol de Kubernetes, defina primero esos permisos como un rol. Los roles de Kubernetes conceden permisos. No hay ningún concepto de denegación de permiso.

Los roles se usan para conceder permisos en un espacio de nombres. Si necesita conceder permisos en todo el clúster o para recursos de clúster fuera de un espacio de nombres determinado, puede usar en su lugar ClusterRoles.

Un ClusterRole funciona de la misma manera para conceder permisos a los recursos, pero se puede aplicar a los recursos de todo el clúster, no solo a los de un espacio de nombres específico.

RoleBindings y ClusterRoleBindings

Una vez que se definen roles para conceder permisos a los recursos, asigne esos permisos de RBAC de Kubernetes con un RoleBinding. Si el clúster de AKS se integra con Azure Active Directory, los enlaces son la solución para conceder permisos a esos usuarios de Azure AD para que realicen acciones en el clúster.

Los enlaces de roles se usan para asignar roles para un espacio de nombres determinado. Este enfoque le permite separar lógicamente un único clúster de AKS, con usuarios que solo pueden acceder a los recursos de la aplicación en su espacio de nombres asignado. Si necesita enlazar roles en todo el clúster o para recursos de clúster fuera de un espacio de nombres determinado, puede usar en su lugar ClusterRoleBindings.

Un ClusterRoleBinding funciona de la misma manera para enlazar roles con usuarios, pero se puede aplicar a los recursos de todo el clúster, no solo a los de un espacio de nombres específico. Este enfoque le permite conceder acceso para los administradores o ingenieros de soporte técnico a todos los recursos del clúster de AKS.

Secretos de Kubernetes

Un secreto de Kubernetes se usa para insertar datos confidenciales en pods, como credenciales de acceso o claves. En primer lugar, cree un secreto mediante la API de Kubernetes. Al definir el pod o la implementación, puede solicitar un secreto específico. Los secretos solo se proporcionan a los nodos que tienen un pod programado que lo requiere, y el secreto se almacena en tmpfs, no se escribe en el disco. Cuando se elimina el último pod en un nodo que requiere un secreto, el secreto se elimina del sistema tmpfs del nodo. Los secretos se almacenan en un espacio de nombres determinado y solo son accesibles para los pods del mismo espacio de nombres.

El uso de secretos reduce la información confidencial que se define en el pod o el manifiesto YAML del servicio. En su lugar, solicite el secreto almacenado en el servidor de API de Kubernetes como parte de su manifiesto YAML. Este enfoque proporciona acceso al secreto solo al pod específico. Tenga en cuenta que los archivos de manifiesto de secreto sin formato contienen los datos secretos en formato base64. Por lo tanto, este archivo debe considerarse como información confidencial y no confirmarse en el control de código fuente.

Contenedores de Windows

Los secretos se escriben en texto no cifrado en el volumen del nodo (a diferencia de Linux, donde se escriben en tmpfs/en memoria). Esto significa que los clientes tienen que hacer dos cosas:

- Usar las ACL de archivo para proteger la ubicación del archivo de secretos
- Usar el cifrado de nivel de volumen mediante BitLocker

Azure Kubernetes Service

AKS es el servicio gestionado por Microsoft Azure para este tipo de clústers. Lo mejor de todo es que te evita tener que realizar toda la instalación De hecho, **los master nodes son completamente administrados por Azure**. Es más, ni siquiera pagas por ellos, ya que **solo pagas por los worker nodes que quieras tener**, los cuales son maquinas virtuales que son configuradas automáticamente para unirse a tu clúster.
Otra de las ventajas que tiene AKS es que simplifica la actualización de tu clúster a nuevas versiones de Kubernetes: lo único que debes hacer es indicar la hora a la que quieres que se lleve a cabo la actualización.

Creación de un clúster en AKS

```
#Create an azure-cli container
docker run -it --rm microsoft/azure-cli sh
#Login
az login
#Select your subscription account
az account set -s "Microsoft Azure Internal Consumption"
#Create a resource group
RESOURCE_GROUP="AKS-Demo"
$RESOURCE_GROUP="AKS-Demo"
LOCATION="northeurope"
az group create -n ${RESOURCE_GROUP} -l ${LOCATION}
#Create a cluster
AKS_NAME="gisaks"
$AKS_NAME="gisaks"
az aks create -g ${RESOURCE_GROUP} -n ${AKS_NAME} \
--node-count 1 --generate-ssh-keys
#Install kubectl if you don't have it
az aks install-cli
#configure kubectl to comunicate with out AKS cluster
az aks get-credentials -g ${RESOURCE_GROUP} -n ${AKS_NAME}
#Check kubectl version
kubectl version --short
kubectl get nodes
kubectl get services --all-namespaces
#Access Kubernetes Dashboard
az aks browse -g ${RESOURCE_GROUP} -n ${AKS_NAME}
#Giving permissions
kubectl create clusterrolebinding kubernetes-dashboard -n kube-system --
clusterrole=cluster-admin --serviceaccount=kube-system:kubernetes-dashboard
#Scale cluster
az aks scale -g ${RESOURCE_GROUP} -n ${AKS_NAME} --node-count 3
#delete the resource group and the cluster
az group delete -n ${RESOURCE_GROUP}
```

Com

Uso de Azure CLI y Powershell para crear un clúster de Kubernetes en Azure

Una vez que haya instalado Powershell y la CLI de Azure, ejecute

az aks install-cli
para instalar el cliente kubectl. Siga las instrucciones para configurar la variable de entorno en el sistema.

Abra VSCode o cualquier otro editor de texto y comience a crear su secuencia de comandos.

Primero lo primero..

```
#login to your azure subscripiton
az login

#see your subscriptions
az account list

#set active subscription if you have more than one
az account set --name "<name of your azure subscripiton>"
```

Primero definimos las variables que necesitamos usar. cambie el valor de las variables como mejor le parezca. Estoy eligiendo la ubicación "Europa del Norte" ya que estoy en Noruega, pero una parte importante es que su nombre de AKS debe ser único dentro de la ubicación/región que elija y, si planea implementar ACS, también su nombre debe ser único a nivel mundial.

```
# general
$location = "northeurope"
$aksrg = "cluster_rg"

# VNET##
$vnet = "cluster-vnet"
$vnet_address_range = "10.0.0.0/16"

# aks subnet
$akssn = "aks-sn"
$aks_cidr = "10.0.2.0/23"

# aks
$aks = "<define a unique name for your cluster (unique within $location)>"
$aks_srv_cidr = "192.168.6.0/23"
$aks_dns_ip = "192.168.6.10"
$aks_pod_cidr = "192.168.8.0/21"
$aks_docker_bridge = "172.17.0.1/16"

# acr
$registry = "<define a globally unique name for your registry>"
```

A continuación crearemos algunos recursos. Comenzamos con nuestro grupo de recursos y luego creamos la red virtual y la subred.

```
# create the resource group
az group create --name $aksrg --location $location
```

```
# define a VNET
az network vnet create --name $vnet --resource-group $aksrg --address-prefixes
$vnet_address_range

# create the AKS subnet inside the vnet
az network vnet subnet create -g $aksrg --vnet-name $vnet -n $akssn --address-prefixes
$aks_cidr

# get id of the vnet
$vnetid = $(az network vnet show -n $vnet -g $aksrg --query id --output tsv)

# get the id of the subnet
$akssnid = $(az network vnet subnet show -g $aksrg --vnet-name $vnet -n $akssn --query id
--output tsv)
```

También necesitamos crear una entidad de servicio que permita que el servicio AKS interactúe con la red virtual que creamos. Un consejo que tengo mientras trabajo con Azure CLI y powershell es instalar "jq". Es un procesador JSON de línea de comandos ligero y flexible que se puede instalar a través de Chocolatey . Todos los comandos de la CLI de Azure pueden generar valores de respuesta json aplicando el modificador "--output json" al comando.

Si desea que su principal de servicio sea un poco más fácil de detectar en AAD, puede adjuntar un nombre al SP mediante el modificador --name (es decir: --name "${aks}_sp"). Además, si adjunta el nombre a la entidad principal del servicio y no lo elimina en la limpieza, se reutilizará y parcheará la próxima vez que ejecute el script.

```
#create the service prinsipal and store the output in a json file
$outputfile = "aks_sp.json"
$filecontent = $(az ad sp create-for-rbac --skip-assignment --name "${aks}_sp" --output
json)
$Utf8NoBomEncoding = New-Object System.Text.UTF8Encoding $False
[System.IO.File]::WriteAllLines($outputfile, $filecontent, $Utf8NoBomEncoding)


#read the values from the json file
$aks_appid = $(jq -r ".appId" $outputfile)
$aks_secret = $(jq -r ".password" $outputfile)
```

"jq" es un poco quisquilloso con la codificación de archivos, así que para asegurarnos de que esté almacenado en un formato compatible, podemos forzarlo a UTF8. En mi computadora portátil, el shell integrado de VScode lo almacena como UTF-16 LE cuando solo descargo el az ad sp create-for-rbac con "> $ archivo de salida"

Con la entidad de servicio creada y cargada en las variables mediante jq, podemos asignar la función de colaborador a la red virtual. Observe también la conversión UTF8. Mientras

estuve probando, descubrí que jq es un poco hambriento sobre cómo le gustan sus archivos codificados. Utf8 funciona bien, por lo que el convertidor solo fuerza eso.

¡NOTA!
Si ejecuta todo esto como una sola secuencia de comandos, lo más probable es que reciba un error de que "Principal <guid> no existe en el directorio <guid>". Un "pequeño truco" es dejar que el script entre en reposo durante unos 10 segundos para que AAD tenga tiempo de crear la entidad de servicio. Todavía no he encontrado una mejor manera de hacer esto, así que si alguien tiene un comando "espere" que funcione con los comandos "az ad", hágamelo saber en los comentarios :)

#ZzzzZZz
Start-Sleep -s 10
az role assignment create --assignee $aks_appid --scope $vnetid --role Contributor
Ahora todo está configurado y podemos crear Azure Kubernetes Service. Otro servicio que puede resultar útil para implementar con AKS es un registro de contenedor. Especialmente cuando uso esto como parte de la configuración de un CI/CD

optional create Azure Container Registry
az acr create --name $registry --resource-group $aksrg --sku basic
Si no creó un ACR, elimine el parámetro --attach-acr. ACR también se puede adjuntar más tarde.

¡NOTA!
Este es un clúster pequeño, sin escalado automático y sin un plan real para la creación de redes. Para crear un clúster listo para producción, necesita un poco más de planificación en torno a IP/red, etc.

#create the aks service (remove linebreaks before running
az aks create --resource-group $aksrg \
    --name $aks \
    --attach-acr $registry \
    --node-count 3 \
    --network-plugin kubenet \
    --service-cidr $aks_srv_cidr \
    --dns-service-ip $aks_dns_ip \
    --pod-cidr $aks_pod_cidr \
    --docker-bridge-address $aks_docker_bridge \
    --vnet-subnet-id $akssnid \
    --service-principal $aks_appid \
    --client-secret $aks_secret

get the aks resource id

$aks_resourceId = $(az aks show -n $aks -g $aksrg --query id -o tsv)
wait for aks to be up and running
az resource wait --exists --ids $aks_resourceId
Ahora el clúster está en funcionamiento. Para establecer el contexto de kubectl en su clúster recién creado, ejecute.

az aks get-credentials --resource-group $aksrg --name $aks --overwrite-existing
Para verificar que todo esté en funcionamiento, obtenga la información del clúster mediante kubectl

kubectl cluster-info
Es importante comprender que este no es un servicio diseñado y protegido para la producción, por lo que debe dedicar más tiempo a la personalización y agregar seguridad, redes y planificación de la capacidad antes de implementar algo para la producción.

Cuando haya terminado, ejecutar dos comandos simples eliminará todo lo que hemos creado.

#cleanup
az group delete --name $aksrg --yes
az group delete --name "NetworkWatcherRG" --yes
az ad sp delete --id $aks_appid
Remove-Item –path $outputfile
Hacer esto en bash debería requerir cambios mínimos. Todos los comandos de la CLI de Azure son iguales para varias plataformas

Inicio rápido: implementar un clúster de Azure Kubernetes Service mediante la CLI de Azure

Azure Kubernetes Service (AKS) es un servicio de Kubernetes administrado que le permite implementar y administrar clústeres rápidamente. En este inicio rápido, usted:

- Implemente un clúster de AKS mediante la CLI de Azure.
- Ejecute una aplicación de varios contenedores de muestra con un front-end web y una instancia de Redis en el clúster.

Este inicio rápido supone una comprensión básica de los conceptos de Kubernetes. Para obtener más información, consulte Conceptos básicos de Kubernetes para Azure Kubernetes Service (AKS) .

Si no tiene una suscripción de Azure , cree una cuenta gratuita de Azure antes de comenzar.

Para obtener más información sobre cómo crear un grupo de nodos de Windows Server, consulte Crear un clúster de AKS que admita contenedores de Windows Server .

requisitos previos

- Use el entorno Bash en Azure Cloud Shell . Para obtener más información, consulte Inicio rápido de Azure Cloud Shell: Bash .

- Si prefiere ejecutar los comandos de referencia de la CLI localmente, instale la CLI de Azure. Si ejecuta Windows o macOS, considere ejecutar la CLI de Azure en un contenedor de Docker. Para obtener más información, consulte Cómo ejecutar la CLI de Azure en un contenedor de Docker .
  - Si usa una instalación local, inicie sesión en la CLI de Azure con el comando az login . Para finalizar el proceso de autenticación, sigue los pasos que se muestran en tu terminal. Para conocer otras opciones de inicio de sesión, consulte Inicio de sesión con la CLI de Azure .
  - Cuando se le solicite, instale la extensión de la CLI de Azure la primera vez que la use. Para obtener más información acerca de las extensiones, consulte Uso de extensiones con la CLI de Azure .
  - Ejecute az version para encontrar la versión y las bibliotecas dependientes que están instaladas. Para actualizar a la última versión, ejecute az upgrade .

- Este artículo requiere la versión 2.0.64 o posterior de la CLI de Azure. Si usa Azure Cloud Shell, la última versión ya está instalada.
- La identidad que está utilizando para crear su clúster tiene los permisos mínimos apropiados. Para obtener más detalles sobre el acceso y la identidad de AKS, consulte Opciones de acceso e identidad para Azure Kubernetes Service (AKS) .
- Si tiene varias suscripciones de Azure, seleccione el ID de suscripción adecuado en el que se deben facturar los recursos mediante el comando az account .
- Verifique que los proveedores de *Microsoft.OperationsManagement* y *Microsoft.OperationalInsights* estén registrados en su suscripción. Estos son los proveedores de recursos de Azure necesarios para admitir Container Insights . Para verificar el estado del registro, ejecute los siguientes comandos:

CLI de Azure

```
az provider show -n Microsoft.OperationsManagement -o table
az provider show -n Microsoft.OperationalInsights -o table
```

Si no están registrados,
registre *Microsoft.OperationsManagement* y *Microsoft.OperationalInsights* m ediante los siguientes comandos:

CLI de Azure

```
az provider register --namespace Microsoft.OperationsManagement
az provider register --namespace Microsoft.OperationalInsights
```

Nota

Ejecute los comandos con privilegios administrativos si planea ejecutar los comandos de este inicio rápido localmente en lugar de Azure Cloud Shell.

Crear un grupo de recursos

Un grupo de recursos de Azure es un grupo lógico en el que se implementan y administran los recursos de Azure. Cuando crea un grupo de recursos, se le solicita que especifique una ubicación. Esta ubicación es:

- La ubicación de almacenamiento de los metadatos de su grupo de recursos.
- Dónde se ejecutarán sus recursos en Azure si no especifica otra región durante la creación del recurso.

El siguiente ejemplo crea un grupo de recursos denominado *myResourceGroup* en la ubicación *eastus* .

Cree un grupo de recursos con el comando az group create .

CLI de Azure

Intentalo

```
az group create --name myResourceGroup --location eastus
```

El siguiente ejemplo de salida se parece a la creación correcta del grupo de recursos:

JSONC

```
{
  "id": "/subscriptions/<guid>/resourceGroups/myResourceGroup",
  "location": "eastus",
  "managedBy": null,
  "name": "myResourceGroup",
  "properties": {
    "provisioningState": "Succeeded"
  },
  "tags": null
```

}

Crear un clúster de AKS

Cree un clúster de AKS con el comando az aks create con el parámetro --enable-addons monitoringy --enable-msi-auth-for-monitoringpara habilitar Azure Monitor Container insights con autenticación de identidad administrada (versión preliminar). El siguiente ejemplo crea un clúster llamado *myAKSCluster* con un nodo y habilita una identidad administrada asignada por el sistema:

CLI de Azure

Intentalo

```
az aks create -g myResourceGroup -n myAKSCluster --enable-managed-identity --node-count 1 --enable-addons monitoring --enable-msi-auth-for-monitoring  --generate-ssh-keys
```

Después de unos minutos, el comando se completa y devuelve información en formato JSON sobre el clúster.

Nota

Cuando crea un clúster de AKS, se crea automáticamente un segundo grupo de recursos para almacenar los recursos de AKS. Para obtener más información, consulte **¿Por qué se crean dos grupos de recursos con AKS?**

Conéctese al clúster

Para administrar un clúster de Kubernetes, use el cliente de línea de comandos de Kubernetes, kubectl . kubectlya está instalado si usa Azure Cloud Shell.

1. Instale kubectllocalmente usando el comando az aks install-cli :

   CLI de Azure

   ```
   az aks install-cli
   ```

2. Configure kubectlpara conectarse a su clúster de Kubernetes mediante el comando az aks get-credentials . El siguiente comando:
   - Descarga las credenciales y configura la CLI de Kubernetes para usarlas.
   - Utiliza ~/.kube/configla ubicación predeterminada para el archivo de configuración de Kubernetes . Especifique una ubicación diferente para su archivo de configuración de Kubernetes usando el argumento *--file* .

   CLI de Azure

   Intentalo

   ```
   az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
   ```

3. Verifique la conexión a su clúster mediante el comando kubectl get . Este comando devuelve una lista de los nodos del clúster.

   CLI de Azure

   Intentalo

   kubectl get nodes
   El siguiente ejemplo de salida muestra el nodo único creado en los pasos anteriores. Asegúrese de que el estado del nodo sea *Listo* :

   Producción

   ```
   NAME                       STATUS   ROLES   AGE     VERSION
   aks-nodepool1-31718369-0   Ready    agent   6m44s   v1.12.8
   ```

Implementar la aplicación

Un archivo de manifiesto de Kubernetes define el estado deseado de un clúster, como qué imágenes de contenedor ejecutar.

En este inicio rápido, usará un manifiesto para crear todos los objetos necesarios para ejecutar la aplicación Azure Vote . Este manifiesto incluye dos implementaciones de Kubernetes :

- Las aplicaciones de ejemplo de Azure Vote Python.
- Una instancia de Redis.

También se crean dos servicios de Kubernetes :

- Un servicio interno para la instancia de Redis.
- Un servicio externo para acceder a la aplicación Azure Vote desde Internet.

1. Cree un archivo con el nombre azure-vote.yamly cópielo en el siguiente manifiesto.
   - Si usa Azure Cloud Shell, este archivo se puede crear usando code, vio nanocomo si estuviera trabajando en un sistema virtual o físico.

   YAML

   ```
   apiVersion: apps/v1
   kind: Deployment
   metadata:
     name: azure-vote-back
   spec:
     replicas: 1
     selector:
       matchLabels:
         app: azure-vote-back
     template:
   ```

```
    metadata:
      labels:
        app: azure-vote-back
    spec:
      nodeSelector:
        "kubernetes.io/os": linux
      containers:
      - name: azure-vote-back
        image: mcr.microsoft.com/oss/bitnami/redis:6.0.8
        env:
        - name: ALLOW_EMPTY_PASSWORD
          value: "yes"
        resources:
          requests:
            cpu: 100m
            memory: 128Mi
          limits:
            cpu: 250m
            memory: 256Mi
        ports:
        - containerPort: 6379
          name: redis
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-back
spec:
  ports:
  - port: 6379
  selector:
    app: azure-vote-back
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: azure-vote-front
spec:
  replicas: 1
  selector:
    matchLabels:
      app: azure-vote-front
  template:
    metadata:
```

```
    labels:
      app: azure-vote-front
  spec:
    nodeSelector:
      "kubernetes.io/os": linux
    containers:
    - name: azure-vote-front
      image: mcr.microsoft.com/azuredocs/azure-vote-front:v1
      resources:
        requests:
          cpu: 100m
          memory: 128Mi
        limits:
          cpu: 250m
          memory: 256Mi
      ports:
      - containerPort: 80
      env:
      - name: REDIS
        value: "azure-vote-back"
---
apiVersion: v1
kind: Service
metadata:
  name: azure-vote-front
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: azure-vote-front
```

2. Implemente la aplicación con el comando kubectl apply y especifique el nombre de su manifiesto YAML:

   Consola

   ```
   kubectl apply -f azure-vote.yaml
   ```

   El siguiente ejemplo se asemeja a la salida que muestra las implementaciones y los servicios creados correctamente:

   Producción

   ```
   deployment "azure-vote-back" created
   service "azure-vote-back" created
   ```

deployment "azure-vote-front" created
service "azure-vote-front" created

Probar la aplicación

Cuando se ejecuta la aplicación, un servicio de Kubernetes expone el front-end de la aplicación a Internet. Este proceso puede tardar unos minutos en completarse.

Supervise el progreso con el comando kubectl get service con el --watchargumento.

CLI de Azure

Intentalo

kubectl get service azure-vote-front --watch

La salida **EXTERNAL-IP**azure-vote-front para el servicio se mostrará inicialmente como *pendiente* .

Producción

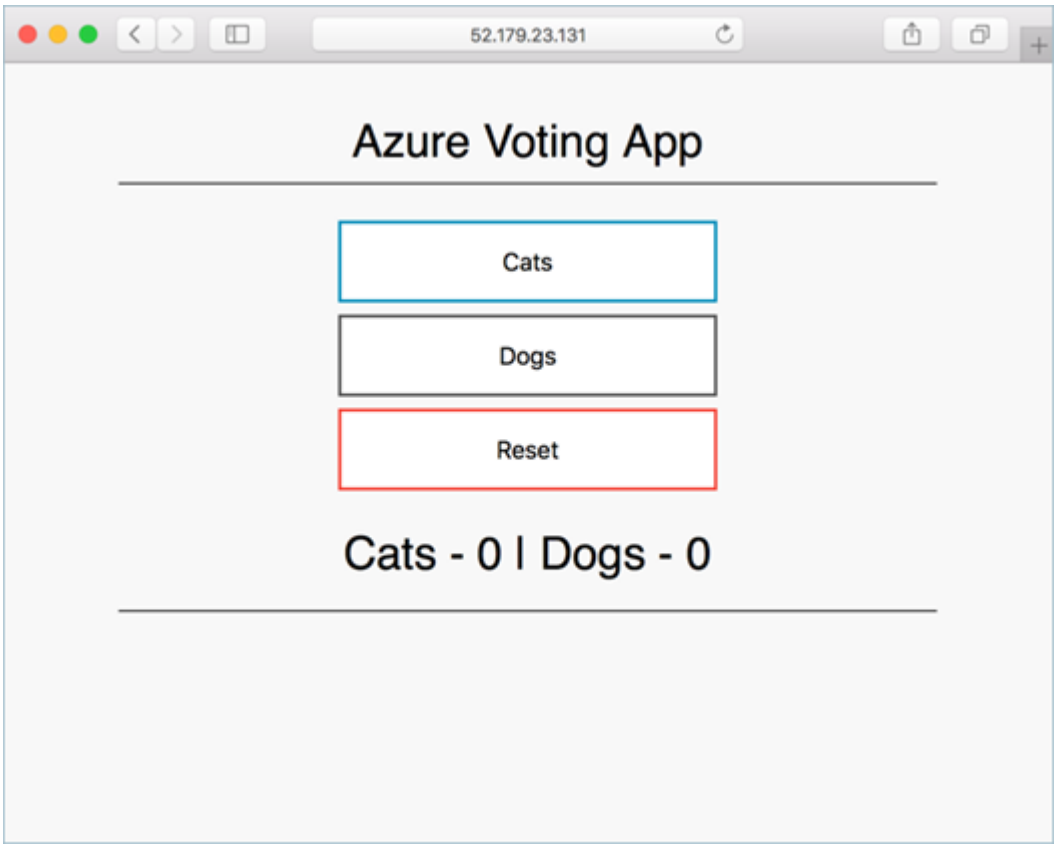
NAME TYPE CLUSTER-IP EXTERNAL-IP PORT(S) AGE
azure-vote-front LoadBalancer 10.0.37.27 <pending> 80:30572/TCP 6s

Una vez que la dirección **IP EXTERNA** cambie de *pendiente* a una dirección IP pública real, use CTRL-Cpara detener el kubectlproceso de observación. El siguiente ejemplo de salida muestra una dirección IP pública válida asignada al servicio:

Producción

azure-vote-front LoadBalancer 10.0.37.27 52.179.23.131 80:30572/TCP 2m

Para ver la aplicación Azure Vote en acción, abra un navegador web en la dirección IP externa de su servicio.



Eliminar el clúster

Para evitar los cargos de Azure, si no planea seguir los tutoriales que siguen, limpie sus recursos innecesarios. Utilice el comando az group delete para eliminar el grupo de recursos, el servicio de contenedor y todos los recursos relacionados.

CLI de Azure

Intentalo

az group delete --name myResourceGroup --yes --no-wait

Nota

El clúster de AKS se creó con una identidad administrada asignada por el sistema (opción de identidad predeterminada utilizada en este inicio rápido), la plataforma administra la identidad y no es necesario quitarla.

Próximos pasos

En este inicio rápido, implementó un clúster de Kubernetes y luego implementó una aplicación simple de varios contenedores en él.

Para obtener más información sobre AKS y recorrer un código completo para el ejemplo de implementación, continúe con el tutorial del clúster de Kubernetes.

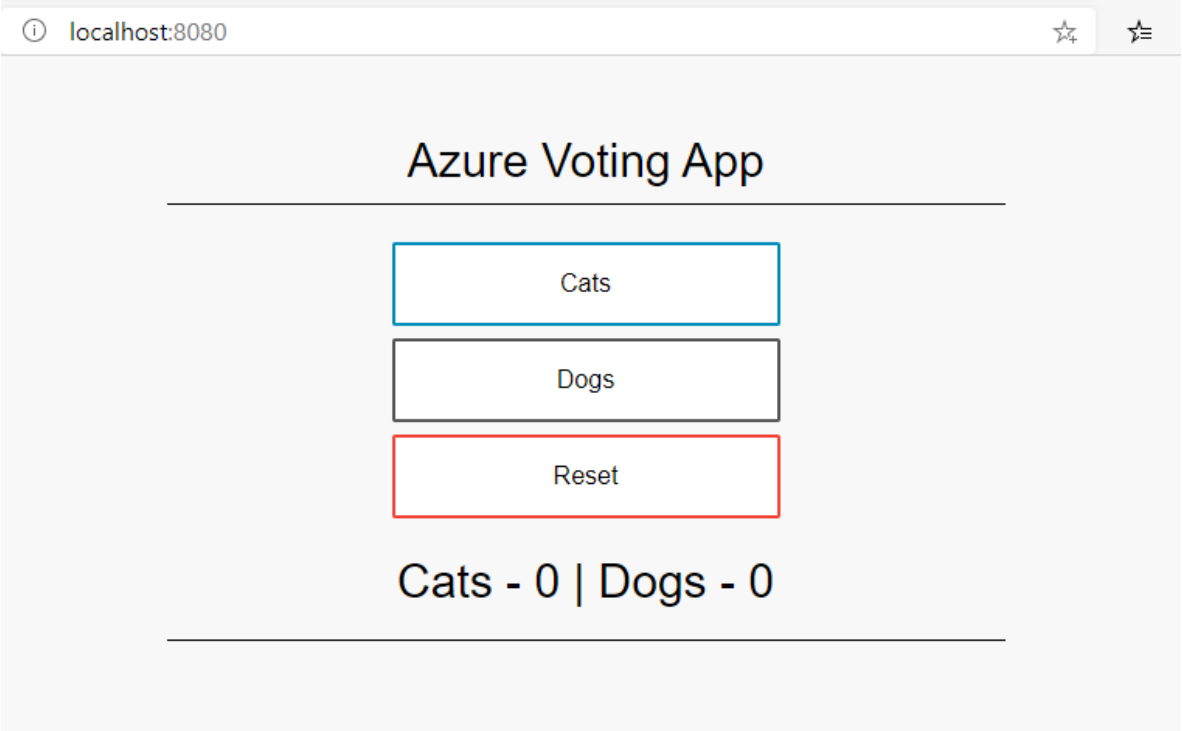
Esta guía de inicio rápido tiene fines introductorios. Para obtener instrucciones sobre cómo crear soluciones completas con AKS para producción, consulte la guía de soluciones de AKS .

Tutorial: preparar una aplicación para Azure Kubernetes Service (AKS)

En este tutorial, la primera parte de siete, se prepara una aplicación de varios contenedores para su uso en Kubernetes. Las herramientas de desarrollo existentes, como Docker Compose, se utilizan para compilar y probar una aplicación localmente. Tu aprendes como:

- Clonar una fuente de aplicación de muestra de GitHub
- Cree una imagen de contenedor a partir de la fuente de la aplicación de muestra
- Pruebe la aplicación de varios contenedores en un entorno Docker local

Una vez completada, la siguiente aplicación se ejecuta en su entorno de desarrollo local:



En tutoriales posteriores, la imagen del contenedor se carga en Azure Container Registry y luego se implementa en un clúster de AKS.

Antes de que empieces

Este tutorial asume una comprensión básica de los conceptos básicos de Docker, como contenedores, imágenes de contenedores y dockercomandos. Para obtener información básica sobre los conceptos básicos de los contenedores, consulte Primeros pasos con Docker .

Para completar este tutorial, necesita un entorno de desarrollo de Docker local que ejecute contenedores de Linux. Docker proporciona paquetes que configuran Docker en un sistema Mac , Windows o Linux .

Nota

Azure Cloud Shell no incluye los componentes de Docker necesarios para completar todos los pasos de estos tutoriales. Por lo tanto, recomendamos utilizar un entorno de desarrollo completo de Docker.

Obtener código de aplicación

La aplicación de muestra utilizada en este tutorial es una aplicación de votación básica que consta de un componente web front-end y una instancia de Redis de back-end. El componente web se empaqueta en una imagen de contenedor personalizada. La instancia de Redis usa una imagen sin modificar de Docker Hub.

Use git para clonar la aplicación de muestra en su entorno de desarrollo:

Consola

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

Cambie al directorio clonado.

Consola

```
cd azure-voting-app-redis
```

Dentro del directorio se encuentra el código fuente de la aplicación, un archivo de composición de Docker creado previamente y un archivo de manifiesto de Kubernetes. Estos archivos se utilizan en todo el conjunto de tutoriales. El contenido y la estructura del directorio son los siguientes:

Producción

```
azure-voting-app-redis
|   azure-vote-all-in-one-redis.yaml
|   docker-compose.yaml
|   LICENSE
|   README.md
|
├───azure-vote
|   |   app_init.supervisord.conf
|   |   Dockerfile
|   |   Dockerfile-for-app-service
|   |   sshd_config
|   |
|   └───azure-vote
|       |   config_file.cfg
```

```
|       |   main.py
|       |
|       ├──static
|       |      default.css
|       |
|       └────templates
|              index.html
|
└────jenkins-tutorial
       config-jenkins.sh
       deploy-jenkins-vm.sh
```

Crear imágenes de contenedores

Docker Compose se puede utilizar para automatizar la creación de imágenes de contenedores y la implementación de aplicaciones de varios contenedores.

Utilice el archivo de muestra docker-compose.yamlpara crear la imagen del contenedor, descargue la imagen de Redis e inicie la aplicación:

Consola

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

Cuando haya terminado, use el comando de imágenes acoplables para ver las imágenes creadas. Se han descargado o creado tres imágenes. La imagen *de azure-vote-front* contiene la aplicación front-end y usa la imagen *de nginx-flask* como base. La imagen de *redis* se usa para iniciar una instancia de Redis.

Copiar

```
$ docker images

REPOSITORY                                  TAG              IMAGE ID       CREATED          SIZE
mcr.microsoft.com/azuredocs/azure-vote-front   v1               84b41c268ad9   9 seconds ago    944MB
mcr.microsoft.com/oss/bitnami/redis         6.0.8            3a54a920bb6c   2 days ago       103MB
tiangolo/uwsgi-nginx-flask                  python3.6        a16ce562e863   6 weeks ago      944MB
```

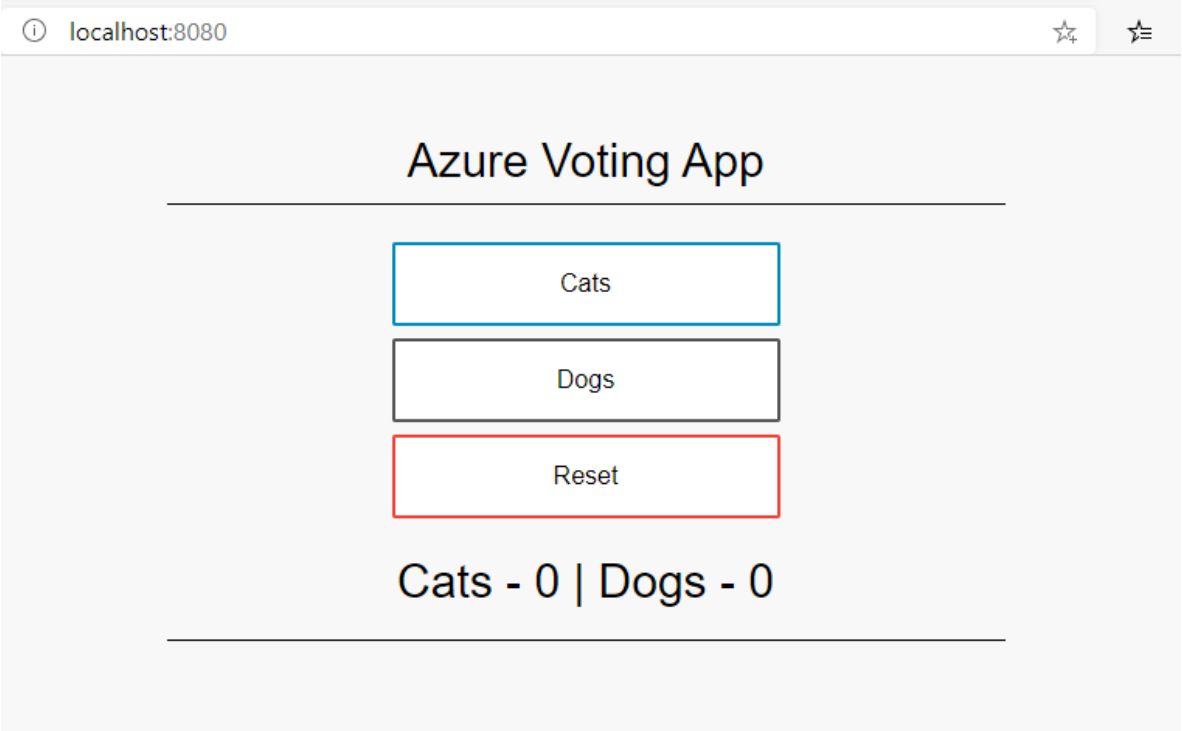
Ejecute el comando docker ps para ver los contenedores en ejecución:

Copiar

```
$ docker ps

CONTAINER ID   IMAGE                                  COMMAND                  CREATED
STATUS          PORTS                    NAMES
```

d10e5244f237 mcr.microsoft.com/azuredocs/azure-vote-front:v1 "/entrypoint.sh /sta…" 3 minutes ago Up 3 minutes 443/tcp, 0.0.0.0:8080->80/tcp azure-vote-front
21574cb38c1f mcr.microsoft.com/oss/bitnami/redis:6.0.8 "/opt/bitnami/script…" 3 minutes ago Up 3 minutes 0.0.0.0:6379->6379/tcp azure-vote-back
Probar la aplicación localmente
Para ver la aplicación en ejecución, ingrese http://localhost:8080en un navegador web local. La aplicación de muestra se carga, como se muestra en el siguiente ejemplo:



Limpiar recursos
Ahora que se ha validado la funcionalidad de la aplicación, los contenedores en ejecución se pueden detener y eliminar. ***No elimine las imágenes del contenedor***: en el siguiente tutorial, la imagen de Azure -*vote-front* se carga en una instancia de Azure Container Registry.

Detenga y elimine las instancias y los recursos del contenedor con el comando docker-compose down :

Consola

docker-compose down
Cuando se eliminó la aplicación local, tiene una imagen de Docker que contiene la aplicación Azure Vote, *azure-vote-front* , para usar con el siguiente tutorial.

Próximos pasos

En este tutorial, se probó una aplicación y se crearon imágenes de contenedor para la aplicación. Aprendiste a:

- Clonar una fuente de aplicación de muestra de GitHub
- Cree una imagen de contenedor a partir de la fuente de la aplicación de muestra
- Pruebe la aplicación de varios contenedores en un entorno Docker local

Avance al siguiente tutorial para obtener información sobre cómo almacenar imágenes de contenedores en Azure Container Registry.

Tutorial: Implementación y uso de Azure Container Registry

Azure Container Registry (ACR) es un registro privado para imágenes de contenedores. Un registro de contenedor privado le permite crear e implementar de forma segura sus aplicaciones y código personalizado. En este tutorial, la segunda parte de siete, implementa una instancia de ACR y le envía una imagen de contenedor. Tu aprendes como:

- Crear una instancia de Azure Container Registry (ACR)
- Etiquetar una imagen de contenedor para ACR
- Sube la imagen a ACR
- Ver imágenes en su registro

En tutoriales posteriores, esta instancia de ACR se integra con un clúster de Kubernetes en AKS y se implementa una aplicación desde la imagen.

Antes de que empieces

En el tutorial anterior , se creó una imagen de contenedor para una aplicación de votación de Azure simple. Si no ha creado la imagen de la aplicación Azure Voting, vuelva al Tutorial 1: Crear imágenes de contenedor .

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az –versión para encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Crear un registro de contenedores de Azure
Para crear un Azure Container Registry, primero necesita un grupo de recursos. Un grupo de recursos de Azure es un contenedor lógico en el que se implementan y administran los recursos de Azure.

- CLI de Azure

Cree un grupo de recursos con el comando az group create . En el siguiente ejemplo, se crea un grupo de recursos denominado *myResourceGroup* en la región *eastus* :
CLI de Azure

```
az group create --name myResourceGroup --location eastus
```

Cree una instancia de Azure Container Registry con el comando az acr create y proporcione su propio nombre de registro. El nombre del registro debe ser único dentro de Azure y contener entre 5 y 50 caracteres alfanuméricos. En el resto de este tutorial, <acrName>se usa como marcador de posición para el nombre del registro del contenedor. Proporcione su propio nombre de registro único. El SKU *básico* es un punto de entrada de costo optimizado para fines de desarrollo que proporciona un equilibrio entre almacenamiento y rendimiento.

CLI de Azure

```
az acr create --resource-group myResourceGroup --name <acrName> --sku Basic
```

Iniciar sesión en el registro de contenedores

- CLI de Azure

Para utilizar la instancia de ACR, primero debe iniciar sesión. Utilice el comando de inicio de sesión az acr y proporcione el nombre exclusivo asignado al registro del contenedor en el paso anterior.
CLI de Azure

```
az acr login --name <acrName>
```

El comando devuelve un mensaje de inicio de *sesión exitoso una vez completado.*

Etiquetar una imagen de contenedor
Para ver una lista de sus imágenes locales actuales, use el comando de imágenes acoplables:

Consola

```
docker images
```

La salida del comando anterior muestra una lista de sus imágenes locales actuales:

Producción

REPOSITORY TAG IMAGE ID CREATED SIZE
mcr.microsoft.com/azuredocs/azure-vote-front v1 84b41c268ad9 7 minutes ago 944MB
mcr.microsoft.com/oss/bitnami/redis 6.0.8 3a54a920bb6c 2 days ago 103MB
tiangolo/uwsgi-nginx-flask python3.6 a16ce562e863 6 weeks ago 944MB

Para usar la imagen del contenedor *azure-vote-front* con ACR, la imagen debe estar etiquetada con la dirección del servidor de inicio de sesión de su registro. Esta etiqueta se usa para el enrutamiento cuando se envían imágenes de contenedores a un registro de imágenes.

- CLI de Azure

Para obtener la dirección del servidor de inicio de sesión, use el comando az acr list y consulte el servidor de inicio de sesión de la *siguiente* manera:

CLI de Azure

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

Ahora, etiquete su imagen local *de Azure-vote-front* con la dirección *acrLoginServer* del registro del contenedor. Para indicar la versión de la imagen, agregue *:v1* al final del nombre de la imagen:

Consola

```
docker tag mcr.microsoft.com/azuredocs/azure-vote-front:v1 <acrLoginServer>/azure-vote-front:v1
```

Para verificar que se apliquen las etiquetas, vuelva a ejecutar las imágenes de la ventana acoplable .

Consola

```
docker images
```

Una imagen se etiqueta con la dirección de instancia de ACR y un número de versión.

Copiar

REPOSITORY TAG IMAGE ID CREATED SIZE
mcr.microsoft.com/azuredocs/azure-vote-front v1 84b41c268ad9 16 minutes ago 944MB
mycontainerregistry.azurecr.io/azure-vote-front v1 84b41c268ad9 16 minutes ago 944MB
mcr.microsoft.com/oss/bitnami/redis 6.0.8 3a54a920bb6c 2 days ago 103MB
tiangolo/uwsgi-nginx-flask python3.6 a16ce562e863 6 weeks ago 944MB

Empujar imágenes al registro

Con su imagen creada y etiquetada, inserte la imagen *de frente de voto azul* en su instancia de ACR. Use docker push y proporcione su propia dirección *acrLoginServer* para el nombre de la imagen de la siguiente manera:

Consola

```
docker push <acrLoginServer>/azure-vote-front:v1
```

Puede tomar algunos minutos completar el envío de la imagen a ACR.

Lista de imágenes en el registro

- CLI de Azure

  Para devolver una lista de imágenes que se enviaron a su instancia de ACR, use el comando az acr repository list . Proporcione el suyo <acrName>de la siguiente manera:

CLI de Azure

```
az acr repository list --name <acrName> --output table
```

El siguiente resultado de ejemplo enumera la imagen *de azure-vote-front* como disponible en el registro:

Producción

```
Result
----------------
azure-vote-front
```

Para ver las etiquetas de una imagen específica, use el comando az acr repository show-tags de la siguiente manera:

CLI de Azure

```
az acr repository show-tags --name <acrName> --repository azure-vote-front --output table
```

El siguiente resultado de ejemplo muestra la imagen *v1* etiquetada en un paso anterior:

Producción

```
Result
--------
v1
```

Ahora tiene una imagen de contenedor almacenada en una instancia privada de Azure Container Registry. Esta imagen se implementa desde ACR en un clúster de Kubernetes en el siguiente tutorial.

Próximos pasos

En este tutorial, creó un Azure Container Registry y envió una imagen para usarla en un clúster de AKS. Aprendiste a:

- Crear una instancia de Azure Container Registry (ACR)

- Etiquetar una imagen de contenedor para ACR
- Sube la imagen a ACR
- Ver imágenes en su registro

Avance al siguiente tutorial para aprender a implementar un clúster de Kubernetes en Azure.

Tutorial: Implementación de un clúster de Azure Kubernetes Service (AKS)

Kubernetes proporciona una plataforma distribuida para aplicaciones en contenedores. Con AKS, puede crear rápidamente un clúster de Kubernetes listo para la producción. En este tutorial, la tercera parte de siete, se implementa un clúster de Kubernetes en AKS. Tu aprendes como:

- Implemente un clúster de Kubernetes AKS que pueda autenticarse en un registro de contenedores de Azure
- Instale la CLI de Kubernetes (kubectl)
- Configure kubectl para conectarse a su clúster de AKS

En tutoriales posteriores, la aplicación Azure Vote se implementa en el clúster, se escala y se actualiza.

Antes de que empieces

En tutoriales anteriores, se creó una imagen de contenedor y se cargó en una instancia de Azure Container Registry. Si no ha realizado estos pasos y le gustaría seguirlos, comience con el Tutorial 1: crear imágenes de contenedores .

- CLI de Azure
-

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az --versionpara encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Crear un clúster de Kubernetes

Los clústeres de AKS pueden usar el control de acceso basado en funciones de Kubernetes (Kubernetes RBAC). Estos controles le permiten definir el acceso a los recursos en función de las funciones asignadas a los usuarios. Los permisos se combinan si a un usuario se le asignan varias funciones, y los permisos se pueden limitar a un solo espacio de nombres o a todo el clúster. De forma predeterminada, la CLI de Azure habilita automáticamente Kubernetes RBAC cuando crea un clúster de AKS.

- CLI de Azure

Cree un clúster de AKS mediante az aks create . El siguiente ejemplo crea un clúster denominado *myAKSCluster* en el grupo de recursos denominado *myResourceGroup* . Este grupo de recursos se creó en el tutorial anterior en la región *eastus* . El siguiente ejemplo no especifica una región, por lo que el clúster de AKS también se crea en la región *eastus* . Para obtener más información, consulte Cuotas, restricciones de tamaño de máquina virtual y disponibilidad regional en Azure Kubernetes Service (AKS) para obtener más información sobre los límites de recursos y la disponibilidad regional para AKS.

Para permitir que un clúster de AKS interactúe con otros recursos de Azure, se crea automáticamente una identidad de clúster, ya que no especificó una. Aquí, esta identidad de clúster tiene derecho a extraer imágenes de la instancia de Azure Container Registry (ACR) que creó en el tutorial anterior. Para ejecutar el comando correctamente, debe tener un rol de **propietario** o **administrador de cuenta de Azure** en la suscripción de Azure.

CLI de Azure

```
az aks create \
    --resource-group myResourceGroup \
    --name myAKSCluster \
    --node-count 2 \
    --generate-ssh-keys \
    --attach-acr <acrName>
```

Para evitar la necesidad de un rol de **propietario** o **administrador de cuenta de Azure** , también puede configurar manualmente una entidad de servicio para extraer imágenes de ACR. Para obtener más información, consulte Autenticación de ACR con entidades principales de servicio o Autenticación desde Kubernetes con un secreto de extracción . Como alternativa, puede usar una identidad administrada en lugar de una entidad de servicio para facilitar la administración.

Después de unos minutos, la implementación se completa y devuelve información en formato JSON sobre la implementación de AKS.

Nota

Para asegurarse de que su clúster funcione de manera confiable, debe ejecutar al menos 2 (dos) nodos.

Instale la CLI de Kubernetes

Para conectarse al clúster de Kubernetes desde su computadora local, use kubectl , el cliente de línea de comandos de Kubernetes.

- CLI de Azure
-

Si usa Azure Cloud Shell, kubectlya está instalado. También puede instalarlo localmente con el comando az aks install-cli :

CLI de Azure

```
az aks install-cli
```

Conéctese al clúster mediante kubectl

- CLI de Azure
-

Para configurar kubectlla conexión a su clúster de Kubernetes, use el comando az aks get-credentials . El siguiente ejemplo obtiene credenciales para el clúster de AKS llamado *myAKSCluster* en *myResourceGroup* :

CLI de Azure

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster
```

Para verificar la conexión a su clúster, ejecute el comando kubectl get nodes para obtener una lista de los nodos del clúster:

CLI de Azure

Intentalo

```
kubectl get nodes
```

El siguiente resultado de ejemplo muestra la lista de nodos de clúster.

Copiar

```
$ kubectl get nodes


NAME                                STATUS   ROLES   AGE     VERSION
aks-nodepool1-37463671-vmss000000   Ready    agent   2m37s   v1.18.10
aks-nodepool1-37463671-vmss000001   Ready    agent   2m28s   v1.18.10
```

Próximos pasos

En este tutorial, se implementó un clúster de Kubernetes en AKS y se configuró kubectlpara conectarse a él. Aprendiste a:

- Implemente un clúster de Kubernetes AKS que pueda autenticarse en un registro de contenedores de Azure
- Instale la CLI de Kubernetes (kubectl)
- Configure kubectl para conectarse a su clúster de AKS

Avance al siguiente tutorial para aprender a implementar una aplicación en el clúster.

Implementar aplicación en Kubernetes

Tutorial: Ejecutar aplicaciones en Azure Kubernetes Service (AKS)

Kubernetes proporciona una plataforma distribuida para aplicaciones en contenedores. Usted crea e implementa sus propias aplicaciones y servicios en un clúster de Kubernetes y deja que el clúster administre la disponibilidad y la conectividad. En este tutorial, la cuarta parte de siete, se implementa una aplicación de muestra en un clúster de Kubernetes. Tu aprendes como:

- Actualizar un archivo de manifiesto de Kubernetes
- Ejecutar una aplicación en Kubernetes
- Probar la aplicación

En tutoriales posteriores, esta aplicación se amplía y actualiza.

Este inicio rápido supone una comprensión básica de los conceptos de Kubernetes. Para obtener más información, consulte Conceptos básicos de Kubernetes para Azure Kubernetes Service (AKS) .

Consejo

Los clústeres de AKS pueden usar GitOps para la administración de la configuración. Esto permite que las declaraciones del estado de su clúster, que se envían al control de código fuente, se apliquen al clúster automáticamente. Para obtener información sobre cómo usar GitOps para implementar una aplicación con un clúster de AKS, consulte el tutorial Use GitOps with Flux v2 y siga los requisitos previos para los clústeres de Azure Kubernetes Service .

Antes de que empieces

En tutoriales anteriores, se empaquetó una aplicación en una imagen de contenedor, esta imagen se cargó en Azure Container Registry y se creó un clúster de Kubernetes.

Para completar este tutorial, necesita el azure-vote-all-in-one-redis.yamlarchivo de manifiesto de Kubernetes creado previamente. Este archivo se descargó con el código fuente de la aplicación en un tutorial anterior. Verifique que haya clonado el repositorio y que haya cambiado los directorios al repositorio clonado. Si no ha realizado estos pasos y le gustaría seguirlos, comience con el Tutorial 1: crear imágenes de contenedores .

- CLI de Azure

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az --versionpara encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Actualizar el archivo de manifiesto

En estos tutoriales, una instancia de Azure Container Registry (ACR) almacena la imagen del contenedor para la aplicación de muestra. Para implementar la aplicación, debe actualizar el nombre de la imagen en el archivo de manifiesto de Kubernetes para incluir el nombre del servidor de inicio de sesión de ACR.

- CLI de Azure

Obtenga el nombre del servidor de inicio de sesión de ACR mediante el comando az acr list de la siguiente manera:

CLI de Azure

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

El archivo de manifiesto de muestra del repositorio de git clonado en el primer tutorial usa las imágenes de Microsoft Container Registry (*mcr.microsoft.com*). Asegúrese de estar en el directorio clonado *de azure-voting-app-redis* , luego abra el archivo de manifiesto con un editor de texto, como vi:

Consola

```
vi azure-vote-all-in-one-redis.yaml
```

Reemplace *mcr.microsoft.com* con su nombre de servidor de inicio de sesión de ACR. El nombre de la imagen se encuentra en la línea 60 del archivo de manifiesto. El siguiente ejemplo muestra el nombre de imagen predeterminado:

YAML

```
containers:

- name: azure-vote-front
```

image: mcr.microsoft.com/azuredocs/azure-vote-front:v1

Proporcione su propio nombre de servidor de inicio de sesión de ACR para que su archivo de manifiesto se parezca al siguiente ejemplo:

YAMLCopiar

containers:

- name: azure-vote-front

image: <acrName>.azurecr.io/azure-vote-front:v1

Guarde y cierre el archivo. En vi, uso :wq.

Implementar la aplicación

Para implementar su aplicación, use el comando kubectl apply . Este comando analiza el archivo de manifiesto y crea los objetos de Kubernetes definidos. Especifique el archivo de manifiesto de muestra, como se muestra en el siguiente ejemplo:

Consola

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

El siguiente resultado de ejemplo muestra los recursos creados correctamente en el clúster de AKS:

Consola

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

deployment "azure-vote-back" created

service "azure-vote-back" created

deployment "azure-vote-front" created

service "azure-vote-front" created

Probar la aplicación

Cuando se ejecuta la aplicación, un servicio de Kubernetes expone el front-end de la aplicación a Internet. Este proceso puede tardar unos minutos en completarse.

Para monitorear el progreso, use el comando kubectl get service con el --watchargumento.

Consola

kubectl get service azure-vote-front --watch

Inicialmente, la *IP EXTERNA* para el servicio *Azure-vote-front* se muestra como *pendiente* :

Producción

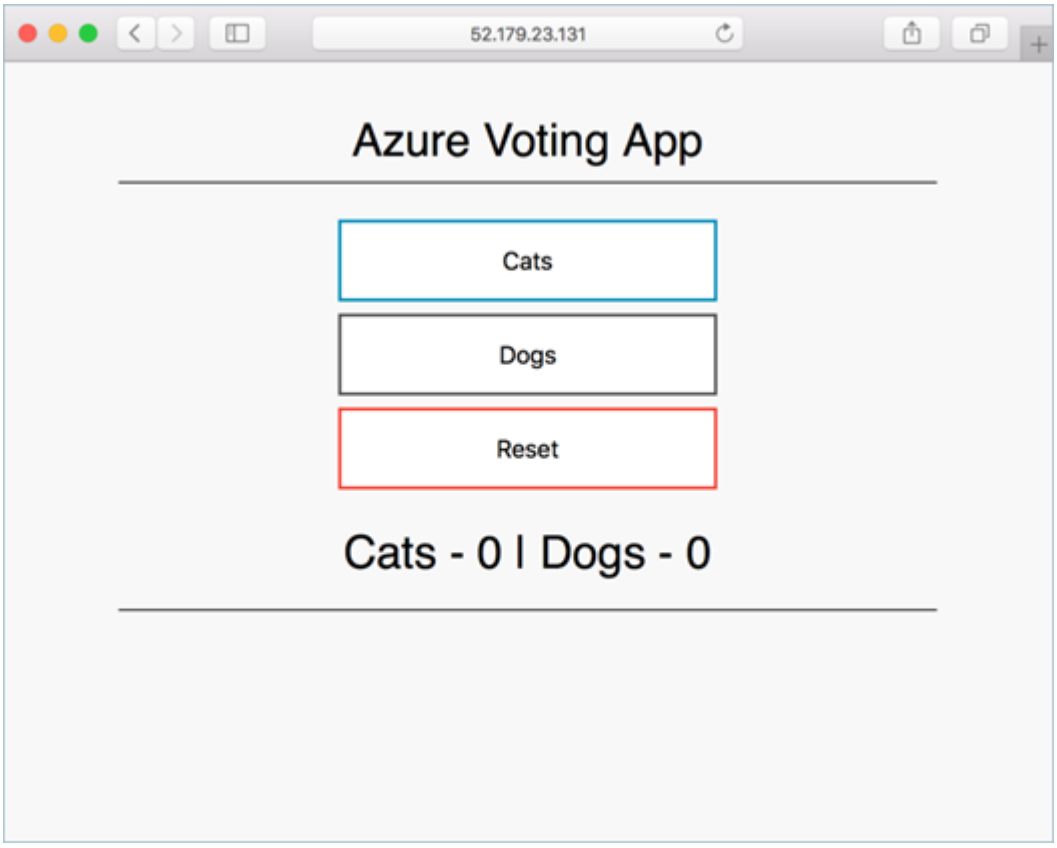
azure-vote-front LoadBalancer 10.0.34.242 <pending> 80:30676/TCP 5s

Cuando la dirección *IP EXTERNA* cambia de *pendiente* a una dirección IP pública real, use CTRL-Cpara detener el kubectlproceso de observación. El siguiente ejemplo de salida muestra una dirección IP pública válida asignada al servicio:

Producción

azure-vote-front LoadBalancer 10.0.34.242 52.179.23.131 80:30676/TCP 67s

Para ver la aplicación en acción, abra un navegador web a la dirección IP externa de su servicio:



Si la aplicación no se cargó, es posible que se deba a un problema de autorización con el registro de imágenes. Para ver el estado de sus contenedores, use el kubectl get podscomando. Si no se pueden extraer las imágenes del contenedor, consulte Autenticación con Azure Container Registry desde Azure Kubernetes Service .

Próximos pasos

En este tutorial, se implementó una aplicación de votación de Azure de muestra en un clúster de Kubernetes en AKS. Aprendiste a:

- Actualizar archivos de manifiesto de Kubernetes
- Ejecutar una aplicación en Kubernetes
- Probar la aplicación

Avance al siguiente tutorial para aprender a escalar una aplicación de Kubernetes y la infraestructura de Kubernetes subyacente.

Tutorial: Escalar aplicaciones en Azure Kubernetes Service (AKS)

Si ha seguido los tutoriales, tiene un clúster de Kubernetes en funcionamiento en AKS e implementó la aplicación de votación de Azure de ejemplo. En este tutorial, parte cinco de siete, escalará los pods en la aplicación y probará el escalado automático de pods. También aprenderá a escalar la cantidad de nodos de máquinas virtuales de Azure para cambiar la capacidad del clúster para hospedar cargas de trabajo. Tu aprendes como:

- Escale los nodos de Kubernetes
- Escale manualmente los pods de Kubernetes que ejecutan su aplicación
- Configurar pods de ajuste de escala automático que ejecutan el front-end de la aplicación

En tutoriales posteriores, la aplicación Azure Vote se actualiza a una nueva versión.

Antes de que empieces

En tutoriales anteriores, una aplicación se empaquetaba en una imagen de contenedor. Esta imagen se cargó en Azure Container Registry y creó un clúster de AKS. Luego, la aplicación se implementó en el clúster de AKS. Si no ha realizado estos pasos y le gustaría seguirlos, comience con el Tutorial 1: crear imágenes de contenedores .

- CLI de Azure

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az --versionpara encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Escalar pods manualmente

Cuando se implementaron el front-end de Azure Vote y la instancia de Redis en tutoriales anteriores, se creó una única réplica. Para ver la cantidad y el estado de los pods en su clúster, use el comando kubectl get de la siguiente manera:

Consola

kubectl get pods

El siguiente resultado de ejemplo muestra un pod de front-end y un pod de back-end:

Producción

NAME READY STATUS RESTARTS AGE

azure-vote-back-2549686872-4d2r5 1/1 Running 0 31m

azure-vote-front-848767080-tf34m 1/1 Running 0 31m

Para cambiar manualmente la cantidad de pods en la implementación *de azure-vote-front* , use el comando kubectl scale . El siguiente ejemplo aumenta la cantidad de pods front-end a *5* :

Consola

kubectl scale --replicas=5 deployment/azure-vote-front

Vuelva a ejecutar kubectl get pods para comprobar que AKS crea correctamente los pods adicionales. Después de aproximadamente un minuto, los pods están disponibles en su clúster:

Consola

kubectl get pods

READY STATUS RESTARTS AGE

azure-vote-back-2606967446-nmpcf 1/1 Running 0 15m

azure-vote-front-3309479140-2hfh0 1/1 Running 0 3m

azure-vote-front-3309479140-bzt05 1/1 Running 0 3m

azure-vote-front-3309479140-fvcvm 1/1 Running 0 3m

azure-vote-front-3309479140-hrbf2 1/1 Running 0 15m

azure-vote-front-3309479140-qphz8 1/1 Running 0 3m

Grupos de escala automática

- CLI de Azure
-

Kubernetes admite el escalado automático de pods horizontales para ajustar la cantidad de pods en una implementación según la utilización de la CPU u otras métricas

seleccionadas. Metrics Server se usa para proporcionar la utilización de recursos a Kubernetes y se implementa automáticamente en los clústeres de AKS versiones 1.10 y posteriores. Para ver la versión de su clúster de AKS, use el comando az aks show , como se muestra en el siguiente ejemplo:

CLI de Azure

```
az aks show --resource-group myResourceGroup --name myAKSCluster --query kubernetesVersion --output table
```

Nota

Si su clúster de AKS es inferior a *1.10* , Metrics Server no se instala automáticamente. Los manifiestos de instalación de Metrics Server están disponibles como un components.yamlactivo en las versiones de Metrics Server, lo que significa que puede instalarlos a través de una URL. Para obtener más información sobre estas definiciones de YAML, consulte la sección Implementación del archivo Léame.

Ejemplo de instalación:

Consola

```
kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/components.yaml
```

Para usar el escalador automático, todos los contenedores en sus pods y sus pods deben tener solicitudes de CPU y límites definidos. En la azure-vote-frontimplementación, el contenedor front-end ya solicita 0,25 CPU, con un límite de 0,5 CPU.

Estas solicitudes y límites de recursos se definen para cada contenedor, como se muestra en el siguiente fragmento de código de ejemplo:

YAML

```
 containers:
 - name: azure-vote-front
   image: mcr.microsoft.com/azuredocs/azure-vote-front:v1
   ports:
   - containerPort: 80
   resources:
     requests:
       cpu: 250m
```

```
    limits:
      cpu: 500m
```

El siguiente ejemplo usa el comando kubectl autoscale para escalar automáticamente la cantidad de pods en la implementación *de azure-vote-front* . Si el uso promedio de la CPU en todos los pods supera el 50 % de su uso solicitado, el escalador automático aumenta los pods hasta un máximo de *10* instancias. A continuación, se define un mínimo de *3 instancias para la implementación:*

Consola

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

Como alternativa, puede crear un archivo de manifiesto para definir el comportamiento del escalador automático y los límites de recursos. El siguiente es un ejemplo de un archivo de manifiesto llamado azure-vote-hpa.yaml.

YAMLCopiar

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: azure-vote-back-hpa
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 3  # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: azure-vote-back
  targetCPUUtilizationPercentage: 50 # target CPU utilization


---


apiVersion: autoscaling/v1
```

```
kind: HorizontalPodAutoscaler
metadata:
  name: azure-vote-front-hpa
spec:
  maxReplicas: 10 # define max replica count
  minReplicas: 3  # define min replica count
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: azure-vote-front
  targetCPUUtilizationPercentage: 50 # target CPU utilization
```

Úselo kubectl applypara aplicar el escalador automático definido en el azure-vote-hpa.yamlarchivo de manifiesto.

Consola

```
kubectl apply -f azure-vote-hpa.yaml
```

Para ver el estado del escalador automático, use el kubectl get hpacomando de la siguiente manera:

Copiar

```
kubectl get hpa
```

```
NAME              REFERENCE                    TARGETS    MINPODS   MAXPODS   REPLICAS   AGE
azure-vote-front  Deployment/azure-vote-front  0% / 50%   3         10        3          2m
```

Después de unos minutos, con una carga mínima en la aplicación Azure Vote, la cantidad de réplicas de pods se reduce automáticamente a tres. Puede usar kubectl get podsde nuevo para ver cómo se eliminan los pods innecesarios.

Nota

Para obtener ejemplos adicionales sobre el uso del escalador automático horizontal de pods, consulte Tutorial de HorizontalPodAutoscaler .

Escale manualmente los nodos de AKS

Si creó su clúster de Kubernetes con los comandos del tutorial anterior, tiene dos nodos. Puede ajustar la cantidad de nodos manualmente si planea más o menos cargas de trabajo de contenedores en su clúster.

El siguiente ejemplo aumenta la cantidad de nodos a tres en el clúster de Kubernetes denominado *myAKSCluster* . El comando tarda un par de minutos en completarse.

CLI de Azure

```
az aks scale --resource-group myResourceGroup --name myAKSCluster --node-count 3
```

Cuando el clúster se ha escalado correctamente, el resultado es similar al siguiente ejemplo:

Producción

```
"agentPoolProfiles": [
 {
  "count": 3,
  "dnsPrefix": null,
  "fqdn": null,
  "name": "myAKSCluster",
  "osDiskSizeGb": null,
  "osType": "Linux",
  "ports": null,
  "storageProfile": "ManagedDisks",
  "vmSize": "Standard_D2_v2",
  "vnetSubnetId": null
 }
```

Próximos pasos

En este tutorial, utilizó diferentes funciones de escalado en su clúster de Kubernetes. Aprendiste a:

- Escale manualmente los pods de Kubernetes que ejecutan su aplicación

- Configurar pods de ajuste de escala automático que ejecutan el front-end de la aplicación
- Escale manualmente los nodos de Kubernetes

Avance al siguiente tutorial para aprender a actualizar la aplicación en Kubernetes.

Tutorial: Actualizar una aplicación en Azure Kubernetes Service (AKS)

Después de implementar una aplicación en Kubernetes, se puede actualizar especificando una nueva imagen de contenedor o versión de imagen. Se organiza una actualización para que solo una parte de la implementación se actualice al mismo tiempo. Esta actualización por etapas permite que la aplicación siga ejecutándose durante la actualización. También proporciona un mecanismo de reversión si se produce un error de implementación.

En este tutorial, la parte seis de siete, se actualiza la aplicación de Azure Vote de ejemplo. Tu aprendes como:

- Actualice el código de la aplicación front-end
- Crear una imagen de contenedor actualizada
- Envíe la imagen del contenedor a Azure Container Registry
- Implementar la imagen de contenedor actualizada

Antes de que empieces

En tutoriales anteriores, una aplicación se empaquetaba en una imagen de contenedor. Esta imagen se cargó en Azure Container Registry y creó un clúster de AKS. Luego, la aplicación se implementó en el clúster de AKS.

También se clonó un repositorio de aplicaciones que incluye el código fuente de la aplicación y un archivo Docker Compose creado previamente que se usa en este tutorial. Verifique que haya creado un clon del repositorio y que haya cambiado los directorios al directorio clonado. Si no completó estos pasos y desea continuar, comience con el Tutorial 1: crear imágenes de contenedores .

- CLI de Azure

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az --versionpara encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Actualizar una aplicación

Realicemos un cambio en la aplicación de muestra y luego actualicemos la versión que ya se implementó en su clúster de AKS. Asegúrese de estar en el directorio *azure-voting-app-redis clonado* . El código fuente de la aplicación de muestra se puede encontrar dentro del directorio *azure-vote* . Abra el archivo *config_file.cfg* con un editor, como vi:

Consola

vi azure-vote/azure-vote/config_file.cfg

Cambie los valores de *VOTE1VALUE* y *VOTE2VALUE* a valores diferentes, como colores. El siguiente ejemplo muestra los valores actualizados:

Copiar

UI Configurations

TITLE = 'Azure Voting App'

VOTE1VALUE = 'Blue'

VOTE2VALUE = 'Purple'

SHOWHOST = 'false'

Guarde y cierre el archivo. En vi, uso :wq.

Actualizar la imagen del contenedor

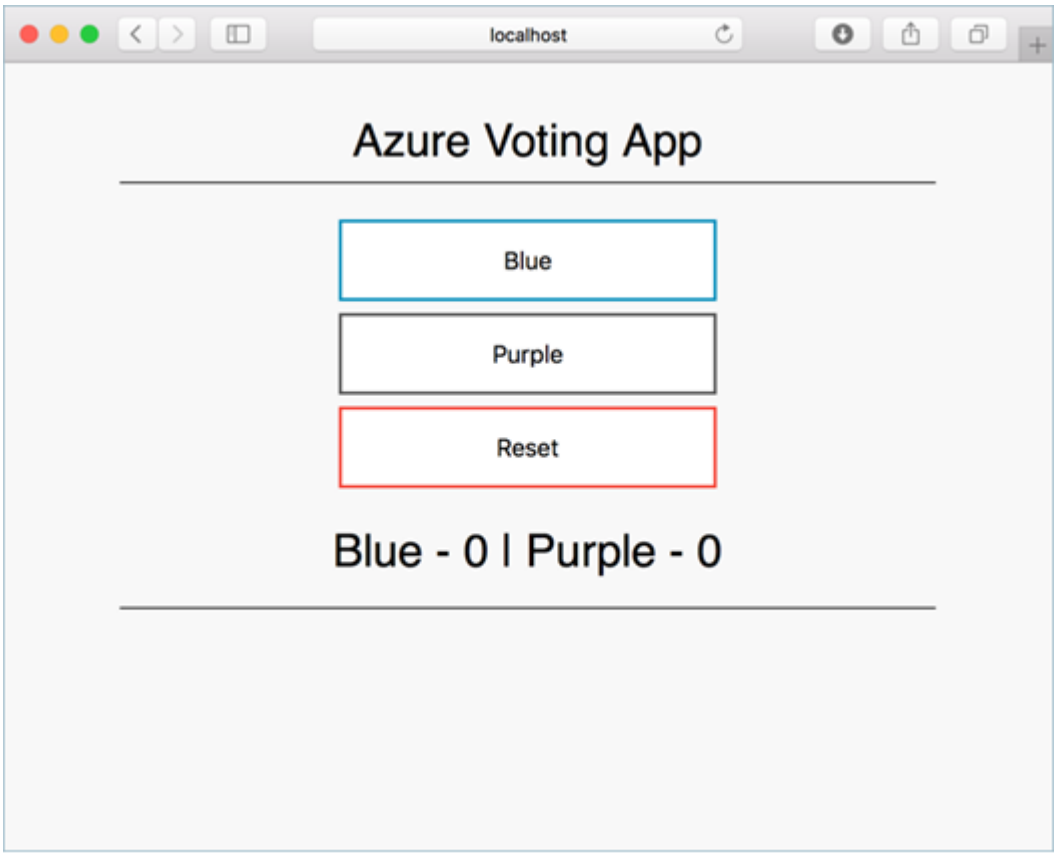
Para volver a crear la imagen de front-end y probar la aplicación actualizada, use docker-compose . El --buildargumento se usa para indicar a Docker Compose que vuelva a crear la imagen de la aplicación:

Consola

docker-compose up --build -d

Probar la aplicación localmente

Para verificar que la imagen del contenedor actualizado muestre sus cambios, abra un navegador web local en http://localhost:8080.



Los valores actualizados proporcionados en el archivo *config_file.cfg* se muestran en su aplicación en ejecución.

Etiquetar y empujar la imagen

- CLI de Azure

Para usar correctamente la imagen actualizada, etiquete la imagen de *azure-vote-front* con el nombre del servidor de inicio de sesión de su registro ACR. Obtenga el nombre del servidor de inicio de sesión con el comando az acr list :

CLI de Azure

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

Use la etiqueta docker para etiquetar la imagen. Reemplácelo <acrLoginServer>con el nombre del servidor de inicio de sesión de ACR o el nombre de host del registro público y actualice la versión de la imagen a *:v2* de la siguiente manera:

Consola

docker tag mcr.microsoft.com/azuredocs/azure-vote-front:v1 <acrLoginServer>/azure-vote-front:v2

Ahora use docker push para cargar la imagen en su registro. Reemplácelo <acrLoginServer>con su nombre de servidor de inicio de sesión de ACR.

CLI de Azure

Nota

Si tiene problemas para ingresar a su registro de ACR, asegúrese de seguir conectado. Ejecute el comando de inicio de sesión az acr con el nombre de su Azure Container Registry que creó en el paso Crear un Azure Container Registry . Por ejemplo, az acr login --name <azure container registry name>.

Consola

docker push <acrLoginServer>/azure-vote-front:v2

Implementar la aplicación actualizada

Para proporcionar el máximo tiempo de actividad, se deben ejecutar varias instancias del módulo de la aplicación. Verifique la cantidad de instancias front-end en ejecución con el comando kubectl get pods :

Copiar

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

azure-vote-back-217588096-5w632 1/1 Running 0 10m

azure-vote-front-233282510-b5pkz 1/1 Running 0 10m

azure-vote-front-233282510-dhrtr 1/1 Running 0 10m

azure-vote-front-233282510-pqbfk 1/1 Running 0 10m

Si no tiene varios pods de front-end, escale la implementación *de azure-vote-front* de la siguiente manera:

Consola

kubectl scale --replicas=3 deployment/azure-vote-front

Para actualizar la aplicación, use el comando kubectl set . Actualice <acrLoginServer>con el servidor de inicio de sesión o el nombre de host de su registro de contenedor y especifique la versión de la aplicación *v2* :

Consola

kubectl set image deployment azure-vote-front azure-vote-front=<acrLoginServer>/azure-vote-front:v2

Para monitorear la implementación, use el comando kubectl get pod . A medida que se implementa la aplicación actualizada, sus pods se terminan y se vuelven a crear con la nueva imagen de contenedor.

Consola

kubectl get pods

El siguiente resultado de ejemplo muestra la finalización de los pods y la ejecución de nuevas instancias a medida que avanza la implementación:

Copiar

$ kubectl get pods

NAME READY STATUS RESTARTS AGE

azure-vote-back-2978095810-gq9g0 1/1 Running 0 5m

azure-vote-front-1297194256-tpjlg 1/1 Running 0 1m

azure-vote-front-1297194256-tptnx 1/1 Running 0 5m

azure-vote-front-1297194256-zktw9 1/1 Terminating 0 1m

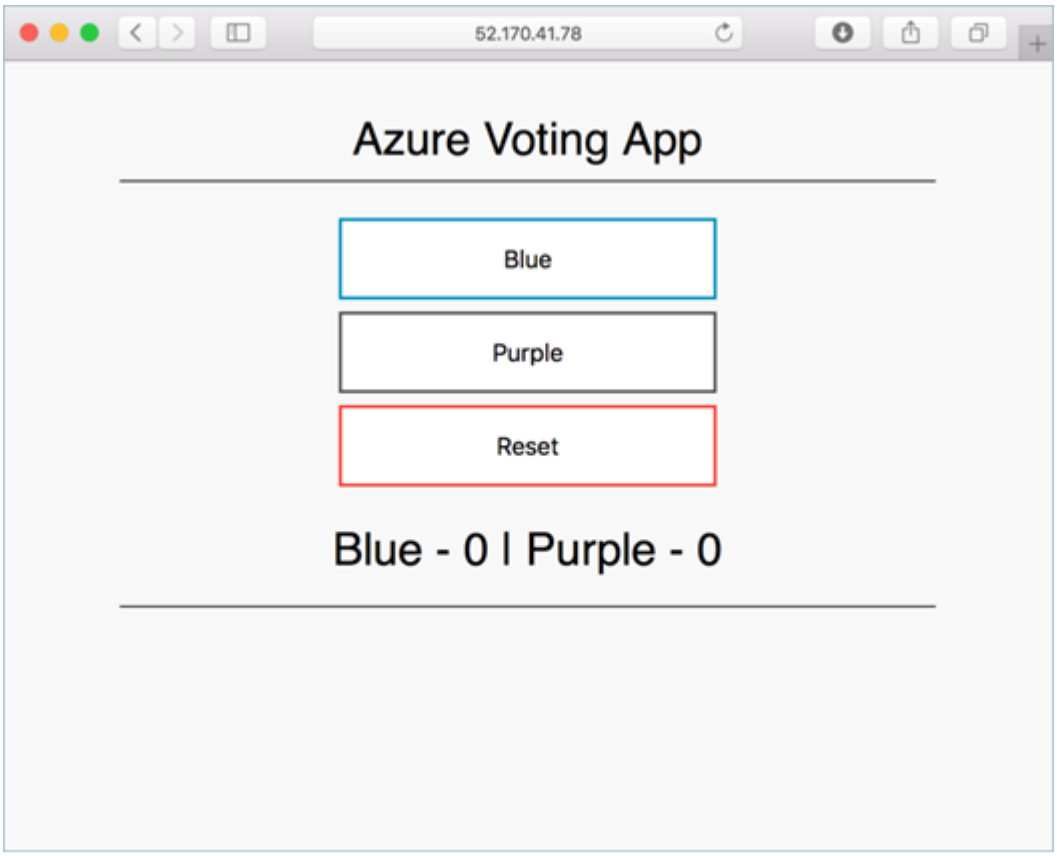
Probar la aplicación actualizada

Para ver la aplicación de actualización, primero obtenga la dirección IP externa del azure-vote-frontservicio:

Consola

kubectl get service azure-vote-front

Ahora abra un navegador web a la dirección IP de su servicio:



Próximos pasos

En este tutorial, actualizó una aplicación e implementó esta actualización en su clúster de AKS. Aprendiste a:

- Actualice el código de la aplicación front-end
- Crear una imagen de contenedor actualizada
- Envíe la imagen del contenedor a Azure Container Registry
- Implementar la imagen de contenedor actualizada

Avance al siguiente tutorial para obtener información sobre cómo actualizar un clúster de AKS a una nueva versión de Kubernetes.

Tutorial: Actualizar Kubernetes en Azure Kubernetes Service (AKS)

Como parte del ciclo de vida de la aplicación y el clúster, es posible que desee actualizar a la última versión disponible de Kubernetes y utilizar nuevas funciones. Un clúster de Azure Kubernetes Service (AKS) se puede actualizar mediante la CLI de Azure.

En este tutorial, parte siete de siete, se actualiza un clúster de Kubernetes. Tu aprendes como:

- Identificar las versiones de Kubernetes actuales y disponibles
- Actualizar los nodos de Kubernetes
- Validar una actualización exitosa

Antes de que empieces

En tutoriales anteriores, una aplicación se empaquetaba en una imagen de contenedor. Esta imagen se cargó en Azure Container Registry y creó un clúster de AKS. Luego, la aplicación se implementó en el clúster de AKS. Si no ha realizado estos pasos y le gustaría seguirlos, comience con el Tutorial 1: crear imágenes de contenedores .

- CLI de Azure

Este tutorial requiere que esté ejecutando la CLI de Azure versión 2.0.53 o posterior. Ejecutar az --versionpara encontrar la versión. Si necesita instalar o actualizar, consulte Instalar la CLI de Azure .

Obtenga las versiones de clúster disponibles

- CLI de Azure

Antes de actualizar un clúster, use el comando az aks get-upgrades para verificar qué versiones de Kubernetes están disponibles para la actualización:

CLI de Azure

```
az aks get-upgrades --resource-group myResourceGroup --name myAKSCluster
```

En el siguiente ejemplo, la versión actual es *1.18.10* y las versiones disponibles se muestran en *actualizaciones* .

JSONCopiar

```
{
  "agentPoolProfiles": null,
  "controlPlaneProfile": {
    "kubernetesVersion": "1.18.10",
    ...
    "upgrades": [
      {
```

```
      "isPreview": null,
      "kubernetesVersion": "1.19.1"
    },
    {
      "isPreview": null,
      "kubernetesVersion": "1.19.3"
    }
  ]
 },
 ...
}
```


Actualizar un clúster

Para minimizar la interrupción de las aplicaciones en ejecución, los nodos de AKS se acordonan y drenan cuidadosamente. En este proceso se realizan los siguientes pasos:

1. El programador de Kubernetes evita que se programen pods adicionales en un nodo que se va a actualizar.
2. Los pods en ejecución en el nodo se programan en otros nodos del clúster.
3. Se crea un nodo que ejecuta los últimos componentes de Kubernetes.
4. Cuando el nuevo nodo está listo y unido al clúster, el programador de Kubernetes comienza a ejecutar pods en él.
5. El nodo anterior se elimina y el siguiente nodo del clúster comienza el proceso de acordonamiento y drenaje.

Nota

Si no se especifica ningún parche, el clúster se actualizará automáticamente al parche GA más reciente de la versión secundaria especificada. Por ejemplo, si se establece --kubernetes-versionen 1.21, el clúster se actualizará a 1.21.9.

Al actualizar por versión secundaria de alias, solo se admite una versión secundaria superior. Por ejemplo, actualizar de 1.20.xa 1.20no activará una actualización al último 1.20parche de GA, pero actualizar a 1.21activará una actualización al último 1.21parche de GA.

- CLI de Azure
-

Use el comando az aks upgrade para actualizar el clúster de AKS.

CLI de Azure

```
az aks upgrade \
   --resource-group myResourceGroup \
   --name myAKSCluster \
   --kubernetes-version KUBERNETES_VERSION
```

Solo puede actualizar una versión secundaria a la vez. Por ejemplo, puede actualizar de *1.14.x* a *1.15.x* , pero no puede actualizar de *1.14.x* a *1.16.x* directamente. Para actualizar de 1.14.xa *1.16.x ,* primero actualice de 1.14.xa *1.15.x ,* luego realice otra actualización de *1.15.xa 1.16.x.*

El siguiente resultado de ejemplo condensado muestra el resultado de la actualización a *1.19.1* . Observe *que kubernetesVersion* ahora informa *1.19.1* :

JSONCopiar


```
{
  "agentPoolProfiles": [
    {
      "count": 3,
      "maxPods": 110,
      "name": "nodepool1",
      "osType": "Linux",
      "storageProfile": "ManagedDisks",
      "vmSize": "Standard_DS1_v2",
    }
  ],
  "dnsPrefix": "myAKSClust-myResourceGroup-19da35",
  "enableRbac": false,
```

"fqdn": "myaksclust-myresourcegroup-19da35-bd54a4be.hcp.eastus.azmk8s.io",

"id": "/subscriptions/<Subscription ID>/resourcegroups/myResourceGroup/providers/Microsoft.ContainerService/managedClusters/myAKSCluster",

"kubernetesVersion": "1.19.1",

"location": "eastus",

"name": "myAKSCluster",

"type": "Microsoft.ContainerService/ManagedClusters"

}

Ver los eventos de actualización

Cuando actualiza su clúster, los siguientes eventos de Kubenetes pueden ocurrir en cada nodo:

- Surge: crea un nodo de oleada.
- Drenaje: los pods se expulsan del nodo. Cada pod tiene un tiempo de espera de 5 minutos para completar el desalojo.
- Actualizar: la actualización de un nodo tuvo éxito o falló.
- Eliminar: eliminó un nodo de aumento.

Úselo kubectl get eventspara mostrar eventos en los espacios de nombres predeterminados mientras ejecuta una actualización. Por ejemplo:

CLI de Azure

Intentalo

```
kubectl get events
```

El siguiente resultado de ejemplo muestra algunos de los eventos anteriores enumerados durante una actualización.

Producción

```
...

default 2m1s Normal Drain node/aks-nodepool1-96663640-vmss000001 Draining node: [aks-nodepool1-96663640-vmss000001]

...
```

default 9m22s Normal Surge node/aks-nodepool1-96663640-vmss000002 Created a surge node [aks-nodepool1-96663640-vmss000002 nodepool1] for agentpool %!s(MISSING)

...

Validar una actualización

- CLI de Azure

Confirme que la actualización se realizó correctamente con el comando az aks show de la siguiente manera:

CLI de Azure

az aks show --resource-group myResourceGroup --name myAKSCluster --output table

El siguiente resultado de ejemplo muestra que el clúster de AKS ejecuta *KubernetesVersion 1.19.1* :

Producción

Name Location ResourceGroup KubernetesVersion ProvisioningState Fqdn

---------- ---------- --------------- ------------------- ------------------- ---------------------------------------
----

myAKSCluster eastus myResourceGroup 1.19.1 Succeeded myaksclust-myresourcegroup-19da35-bd54a4be.hcp.eastus.azmk8s.io

Eliminar el clúster

- CLI de Azure
-

Dado que este tutorial es la última parte de la serie, es posible que desee eliminar el clúster de AKS. Dado que los nodos de Kubernetes se ejecutan en máquinas virtuales (VM) de Azure, continúan incurriendo en cargos incluso si no usa el clúster. Utilice el comando az group delete para eliminar el grupo de recursos, el servicio de contenedor y todos los recursos relacionados.

CLI de Azure

Intentalo

az group delete --name myResourceGroup --yes --no-wait

Nota

Cuando elimina el clúster, la entidad de servicio de Azure Active Directory que usa el clúster de AKS no se elimina. Para conocer los pasos sobre cómo quitar la entidad de servicio, consulte Eliminación y consideraciones de la entidad de servicio de AKS . Si usó una identidad administrada, la identidad es administrada por la plataforma y no requiere que proporcione o rote ningún secreto.

Próximos pasos

En este tutorial, actualizó Kubernetes en un clúster de AKS. Aprendiste a:

- Identificar las versiones de Kubernetes actuales y disponibles
- Actualizar los nodos de Kubernetes
- Validar una actualización exitosa

Para obtener más información sobre AKS, consulte Información general sobre AKS . Para obtener instrucciones sobre cómo crear soluciones completas con AKS, consulte la guía de soluciones de AKS .