

# 使用 Azure AKS 搭建 Kubernetes 基本应用

Azure Kubernetes Service (AKS) 是 Azure 托管的 Kubernetes 环境，使用户无需具备容器业务流程专业知识即可快速、轻松地部署和管理容器化的应用程序。它还通过按需预配、升级和缩放资源，消除了正在进行的操作和维护的负担，而无需使应用程序脱机。

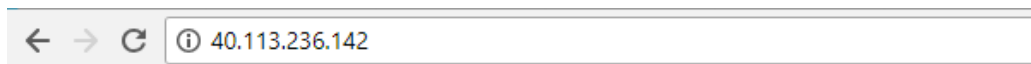
本课程将帮助您了解如何部署 AKS 集群以及在 AKS 群集上部署、管理应用程序，包含以下步骤：

- 1) [为 AKS 准备应用程序及制作镜像](#)
- 2) [创建容器注册表及将映像推送到注册表](#)
- 3) [创建 Azure Kubernetes 服务 \(AKS\) 群集](#)
- 4) [运行应用程序](#)
- 5) [缩放应用程序](#)

## 1 为 AKS 准备应用程序

在本教程的第 1 部分中，将准备一个要在 Kubernetes 中使用的容器应用程序。包含的步骤包括：

- 克隆 GitHub 中的应用程序源
- 根据应用程序源创建容器映像
- 在本地 Docker 环境中测试应用程序

		
<h2>Index of /</h2>		
<hr/>		
<a href="#">../</a>	19-Jun-2018 09:02	-
<a href="#">azurefile/</a>	01-May-2018 13:33	545
<a href="#">genfiles.sh</a>	03-May-2018 07:26	281
<a href="#">gitclone.sh</a>		

在后续教程中，此容器映像会上传到 Azure 容器注册表，然后在 AKS 群集中运行。

### 1.1 开始之前

本教程假定基本了解核心 Docker 的概念，如容器、容器映像和基本的 Docker 命令。如有需要，请参阅 [Docker 入门](#)，了解容器基本知识。

若要完成本教程，需要 Docker 开发环境。Docker 提供的包可在任何 [Mac](#)、[Windows](#) 或 [Linux](#) 系统上轻松配置 Docker。

Azure Cloud Shell 不包含完成本教程每个步骤所需的 Docker 组件。因此，我们建议使用完整的 Docker 开发环境。

## 1.2 获取应用程序代码

本教程使用的示例应用程序是一个 nginx 应用，该应用会搭建一个简单的 nginx web 服务，后端存储使用了 [Azure file](#)。

使用 git 可将应用程序的副本下载到开发环境。

```
git clone https://github.com/andyzhangx/k8s-demo.git
```

目录内包含预创建的 Dockerfile 文件和 配置文件。整套教程都会使用这些文件。

## 1.3 创建容器映像

```
cd k8s-demo/nginx-server/deployment/nginx-server-image/  
sudo docker build --no-cache -t nginx-server:1.0.0 .
```

完成后，使用 [docker images](#) 命令查看创建的映像。

```
sudo docker images
```

输出:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx-server	1.0.0	88bc77129276	3 hours ago	213 MB

## 1.4 在本地测试应用程序

```
sudo docker run -d --name nginx -p 80:80 nginx-server:1.0.0  
curl http://127.0.0.1
```

输出

```
<html>
<head> <title>Index of /</title> </head>
<body bgcolor="white">
<h1>Index of /</h1> <hr> <pre> <a href="..">../</a>
<a href="genfiles.sh">genfiles.sh</a>                01-May-2018 13:33        545
<a href="gitclone.sh">gitclone.sh</a>                03-May-2018 07:26        281
</pre> <hr> </body>
</html>
```

## 1.5 清理资源

现已验证应用程序功能，可停止并删除正在运行的容器。请勿删除容器映像。

运行以下命令，停止并删除正在运行的容器。

```
sudo docker stop nginx && docker rm nginx
```

## 2 创建容器注册表

### 2.1 部署 Azure 容器注册表

在部署 Azure 容器注册表时，首先需要有一个资源组。Azure 资源组是在其中部署和管理 Azure 资源的逻辑容器。

使用 `az group create` 命令创建资源组。在此示例中，在 `eastus` 区域中创建了名为 `myResourceGroup` 的资源组。当前支持的区域包括：`eastus`, `westeurope`, `centralus`, `canadacentral`, `canadaeast`, `uksouth`, `westus`, `westus2`, `australiaeast`, `northeurope`

首先设置变量，以便在后续步骤中使用：

```
RESOURCE_GROUP_NAME=myResourceGroup001 #change here!!!
LOCATION=eastus
```

创建资源组：

```
az group create --name $RESOURCE_GROUP_NAME --location $LOCATION
```

使用 `az acr create` 命令创建 Azure 容器注册表。注册表名称在 Azure 中必须唯一，并且包含 5-50 个字母数字字符。

```
ACR_NAME=acrname01 #change here!!!
```

```
az acr create --resource-group $RESOURCE_GROUP_NAME --name $ACR_NAME --sku Basic
```

## 2.2 容器注册表登录

运行 `az acr login` 命令，登录 ACR 实例。需要提供创建容器注册表时所使用的唯一名称。

```
az acr login --name $ACR_NAME
```

完成后，该命令会返回 “Login Succeeded” 消息。

## 2.3 标记容器映像

若要查看当前映像的列表，请使用 `docker images` 命令。

```
sudo docker images
```

输出：

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx-server	latest	88bc77129276	13 minutes ago	213MB

需要使用注册表的 `loginServer` 名称标记每个容器映像。在将容器映像推送到映像注册表时，使用此标记进行路由。

使用 `az acr show` 命令获取 `loginServer` 名称。

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

输出：

```
AcrLoginServer
```

```
-----  
acrname01.azurecr.io
```

此时，使用容器注册表的 loginServer 标记 `nginx-server` 映像。

```
ACR_SERVER=acrname01.azurecr.io #change here!!!  
sudo docker tag nginx-server:1.0.0 $ACR_SERVER/nginx-server:1.0.0
```

标记后即可运行 `docker images` 验证操作。

```
sudo docker images
```

输出：

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
nginx-server	1.0.0	88bc77129276	8 minutes ago	213 MB
acrname01.azurecr.io/nginx-server	1.0.0	eaf2b9c57e5e	8 minutes ago	213 MB

## 2.4 将映像推送到注册表

将 `nginx-server` 映像推送到注册表。

使用以下示例，将 ACR loginServer 名称替换为环境中的 loginServer。

```
sudo sdocker push $ACR_SERVER/nginx-server:1.0.0
```

此操作需要几分钟时间完成。

## 2.5 列出注册表中的映像

若要返回已推送到 Azure 容器注册表的映像列表，请使用 `az acr repository list` 命令。使用 ACR 实例名称更新命令。

```
az acr repository list --name $ACR_NAME --output table
```

输出：

```
Result
-----
nginx-server
```

然后，若要查看特定映像的标记，请使用 `az acr repository show-tags` 命令。

```
az acr repository show-tags --name $ACR_NAME --repository nginx-server --output table
```

输出：

```
Result
-----
1.0.0
```

完成本教程后，容器映像已存储在专用 Azure 容器注册表实例中。在后续教程中，此映像会从 ACR 部署到 Kubernetes 群集。

## 3 创建 Azure Kubernetes 服务 (AKS) 群集

### 3.1 生成创建 Kubernetes 群集需要的 Service Principal

```
az ad sp create-for-rbac --role="Contributor"
```

输出

```
Retrying role assignment creation: 1/36
{
  "appId": "8dac0304-90bd-4b9d-9f94-b7d3415c3dfg",
  "displayName": "azure-cli-2018-06-20-02-47-03",
  "name": "http://azure-cli-2018-06-20-02-47-03",
  "password": "此处省略",
  "tenant": "此处省略"
}
```

此处生成的 appId, password 会在下一步中使用到。

## 3.2 配置 ACR 身份验证

需要在 AKS 群集和 ACR 注册表之间配置身份验证。这涉及到授予 AKS 标识相应的权限，以便从 ACR 注册表拉取映像。

首先，拷贝并设置 appId，即上一步中的 appId 值。

```
APP_ID=8dac0304-90bd-4b9d-9f94-b7d3415c3dfg #change here!!!
```

获取 ACR 注册表资源 ID。将注册表名称更新为 ACR 注册表的名称，将资源组更新为 ACR 注册表所在的资源组。

```
ACR_ID=$(az acr show --name $ACR_NAME --resource-group $RESOURCE_GROUP_NAME --query "id" --output tsv)
echo $ACR_ID
```

输出

```
/subscriptions/{subs-id}/resourceGroups/andy-acr/providers/Microsoft.ContainerRegistry/registries/<acrName>
```

创建角色分配，以便授予相应的访问权限。

```
az role assignment create --assignee $APP_ID --role Reader --scope $ACR_ID
```

## 3.3 创建 Kubernetes 群集

下面的示例在 `myResourceGroup` 资源组中创建 `myAKSCluster` 群集。此资源组是在上一教程中创建的。

```
CLUSTER_NAME=myAKSCluster001 #change here!!!
CLIENT_SECRET=此处省略 #change here!!!, 此处填写 3.1 步骤输出中的 password 值
```

```
az aks create --resource-group $RESOURCE_GROUP_NAME --name $CLUSTER_NAME --node-count 1 --
generate-ssh-keys --service-principal $APP_ID --client-secret $CLIENT_SECRET
```

几分钟后，部署完成并返回有关 AKS 部署的 JSON 格式信息。

### 3.4 安装 kubectl CLI

若要从客户端计算机连接到 Kubernetes 群集，请使用 `kubectl`（Kubernetes 命令行客户端）。

如果使用的是 Azure CloudShell，则 `kubectl` 已安装。如果要在本地安装它，请运行以下命令：

```
sudo az aks install-cli
```

### 3.5 连接 kubectl

若要配置 `kubectl` 以连接到 Kubernetes 群集，请运行以下命令：

```
az aks get-credentials --resource-group $RESOURCE_GROUP_NAME --name $CLUSTER_NAME
```

若要验证与群集之间的连接，请运行 `kubectl get nodes` 命令。

```
kubectl get nodes
```

输出：

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-14171628-0	Ready	agent	1h	v1.9.6

教程完成时，AKS 群集已准备就绪，可用于工作负荷。在后续教程中，一个容器应用程序会部署到此群集、进行扩大及缩放。

## 4 运行应用程序

### 4.1 更新清单文件

在本教程中，Azure 容器注册表 (ACR) 用于存储容器映像。运行应用程序前，需要先在 Kubernetes 清单文件中更新 ACR 登录服务器名称。

使用 `az acr list` 命令获取 ACR 登录服务器名称。

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



此清单文件已预创建，其中包含 `andyzhangx` 的登录服务器名称。使用任意文本编辑器打开此文件。

```
wget https://raw.githubusercontent.com/andyzhangx/k8s-demo/master/nginx-server/nginx-server-azurefile.yaml
```

将 `andyzhangx` 替换为 ACR 登录服务器名称。

```
containers:
- name: nginx-server
  image: andyzhangx/nginx-server:1.0.0
```

然后，上述代码将如下所示：

```
containers:
- name: nginx-server
  image: <acrName>.azurecr.io/nginx-server:1.0.0
```

保存并关闭该文件。

## 4.2 部署应用程序

使用 `kubectl apply` 命令运行该应用程序。此命令分析清单文件并创建定义的 Kubernetes 对象。

```
kubectl apply -f nginx-server-azurefile.yaml
```

输出：

```
storageclass "azurefile" created
persistentvolumeclaim "pvc-azurefile" created
deployment "nginx-server" created
service "nginx-server" created
```

## 4.3 测试应用程序

创建向 Internet 公开应用程序的 [Kubernetes 服务](#)。此过程可能需要几分钟。

若要监视进度，请将 `kubectl get service` 命令与 `--watch` 参数配合使用。

```
kubectl get service nginx-server --watch
```

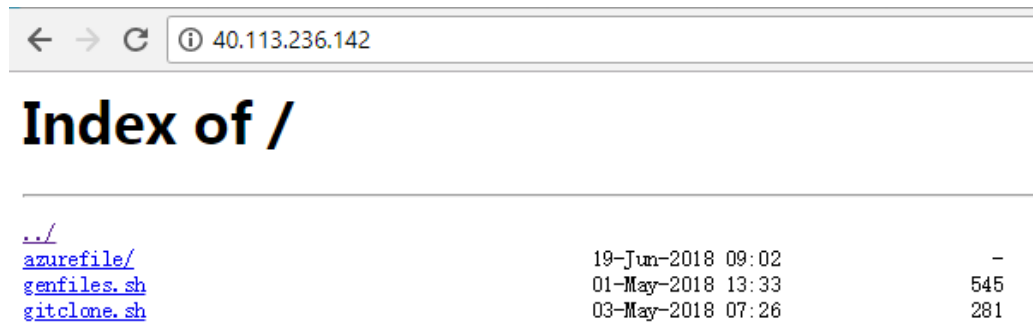
nginx-server 服务的 EXTERNAL-IP 一开始显示为 “pending” 。

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
nginx-server	LoadBalancer	10.0.191.1	<pending>	80:31220/TCP	30s

EXTERNAL-IP 地址从 “pending” 变为 IP 地址以后，请使用 `CTRL-C` 停止 kubectl 监视进程。

```
nginx-server 10.0.34.242 52.179.23.131 80:30676/TCP 2m
```

若要查看应用程序，请浏览到外部 IP 地址。



如果应用程序未加载，可能是因为映像注册表存在授权问题。可以使用如下命令查看 pod 状态：

```
kubectl describe po
```

可以直接使用 docker hub 镜像：

```
kubectl create -f https://raw.githubusercontent.com/andyzhangx/k8s-demo/master/nginx-server/nginx-server-azurefile.yaml
```

## 5 缩放应用程序

### 5.1 手动缩放 Pod

到目前为止，Azure nginx server 实例已部署，且仅有一个副本。若要验证，请运行 [kubectl get](#) 命令。

```
kubectl get pods
```

输出：

NAME	READY	STATUS	RESTARTS	AGE
nginx-server-848767080-tf34m	1/1	Running	0	31m

使用 [kubectl scale](#) 命令手动更改 `nginx-server` 部署中的 Pod 数。此示例将该数量增加到 5。

```
kubectl scale --replicas=5 deployment/nginx-server
```

运行 [kubectl get pods](#) 以验证 Kubernetes 是否在创建 Pod。几分钟左右之后，其他 Pod 在运行：

```
kubectl get pods
```

输出：

NAME	READY	STATUS	RESTARTS	AGE
nginx-server-66bb77c589-7n57x	0/1	ContainerCreating	0	17s
nginx-server-66bb77c589-f74kq	1/1	Running	0	30m
nginx-server-66bb77c589-kfcqd	0/1	ContainerCreating	0	17s
nginx-server-66bb77c589-m2tm6	1/1	Running	0	17s
nginx-server-66bb77c589-w8dfc	1/1	Running	0	17s

### 5.2 缩放 AKS 节点

如果在前面的教程中使用命令创建了 Kubernetes 群集，则它具有一个节点。如果计划在群集上有更多或更少的容器工作负荷，则可以手动调整节点数。

下面的示例将名为 myAKSCluster 的 Kubernetes 群集中的节点数增加到 3 个。该命令需要几分钟时间完成。

```
az aks scale --resource-group=$RESOURCE_GROUP_NAME --name=$CLUSTER_NAME --node-count 3
```

输出类似于：

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

再次运行 `kubectl get nodes` 命令。

```
kubectl get nodes
```

输出：

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-14171628-0	Ready	agent	2h	v1.9.6
aks-nodepool1-14171628-1	Ready	agent	6m	v1.9.6
aks-nodepool1-14171628-2	Ready	agent	6m	v1.9.6

以上步骤中的所有命令，都可以从这里找到：

<https://github.com/andyzhangx/k8s-demo/blob/master/nginx-server/2018LC3-hands-on-lab.md>

## 6 FAQ

### 1. Azure 中国版与国际版有什么区别？

中国版由世纪互联运营，与国际版逻辑上独立。

### 2. AKS 是什么

Azure Kubernetes Service, Azure 提供的托管的 Kubernetes 服务，参考：

<https://azure.microsoft.com/zh-cn/services/kubernetes-service/>

### 3. AKS 和 ACR 在 Azure 已经正式可用了吗？

AKS 国际版已经正式可用

ACR 国际版已经正式可用，中国版正在内部测试。

### 4. ACR 的价格？

ACR 的基本费用按日计算，如果使用附加服务（webhook, container builder）将额外计费

参考：<https://azure.microsoft.com/en-us/pricing/details/container-registry/>

### 5. AKS 的价格？

使用 AKS 时，Kubernetes 的控制节点(master nodes)由 Azure 托管，并且用户不需要支付费用。

工作节点（agent nodes）运行在用户的订阅中，用户只需按标准的虚拟机服务付费。

参考：<https://azure.microsoft.com/en-us/pricing/details/virtual-machines/linux/>

### 6. 更多 AKS 问题请查看

<https://docs.microsoft.com/en-us/azure/aks/faq> (English)

<https://docs.microsoft.com/zh-cn/azure/aks/faq> (中文)