

Tech Summit 2018

Hands-on lab

使用 AKS 搭建 Kubernetes 应用

目录

目录.....	2
1. 概述.....	4
1.1. 课程目标.....	4
1.2. 系统环境.....	4
1.3. 在虚拟机中访问本文档.....	4
2. 创建 Azure Kubernetes 服务 (AKS) 群集.....	5
2.1. 登陆 Azure 控制台.....	5
2.2. 创建 Kubernetes 群集.....	6
2.3. 连接 kubectl.....	7
3. 为 AKS 准备应用程序.....	7
3.1. 获取应用程序代码.....	8
3.2. 创建容器映像.....	8
3.3. 在本地测试应用程序.....	8
3.4. 清理资源.....	9
4. 创建容器注册表.....	9
3.1 部署 Azure 容器注册表.....	9
3.2 容器注册表登录.....	9
3.3 标记容器映像.....	10
3.4 将映像推送到注册表.....	10
3.5 列出注册表中的映像.....	11
5. 运行应用程序.....	11
5.1 配置 ACR 身份验证.....	12
5.2 更新清单文件.....	13

5.3	部署应用程序.....	13
5.4	测试应用程序.....	14
5.5	(可选步骤) 测试 Azure File 功能.....	15
6	缩放应用程序	18
6.1	手动缩放 Deployment	18
6.2	通过VSCODE 扩展控制缩放 Deployment.....	18
6.3	缩放AKS 节点.....	24
1.	FAQ.....	25

1. 概述

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。

1.1. 课程目标

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，包含以下内容：

- 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](#)

1.2. 系统环境

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。

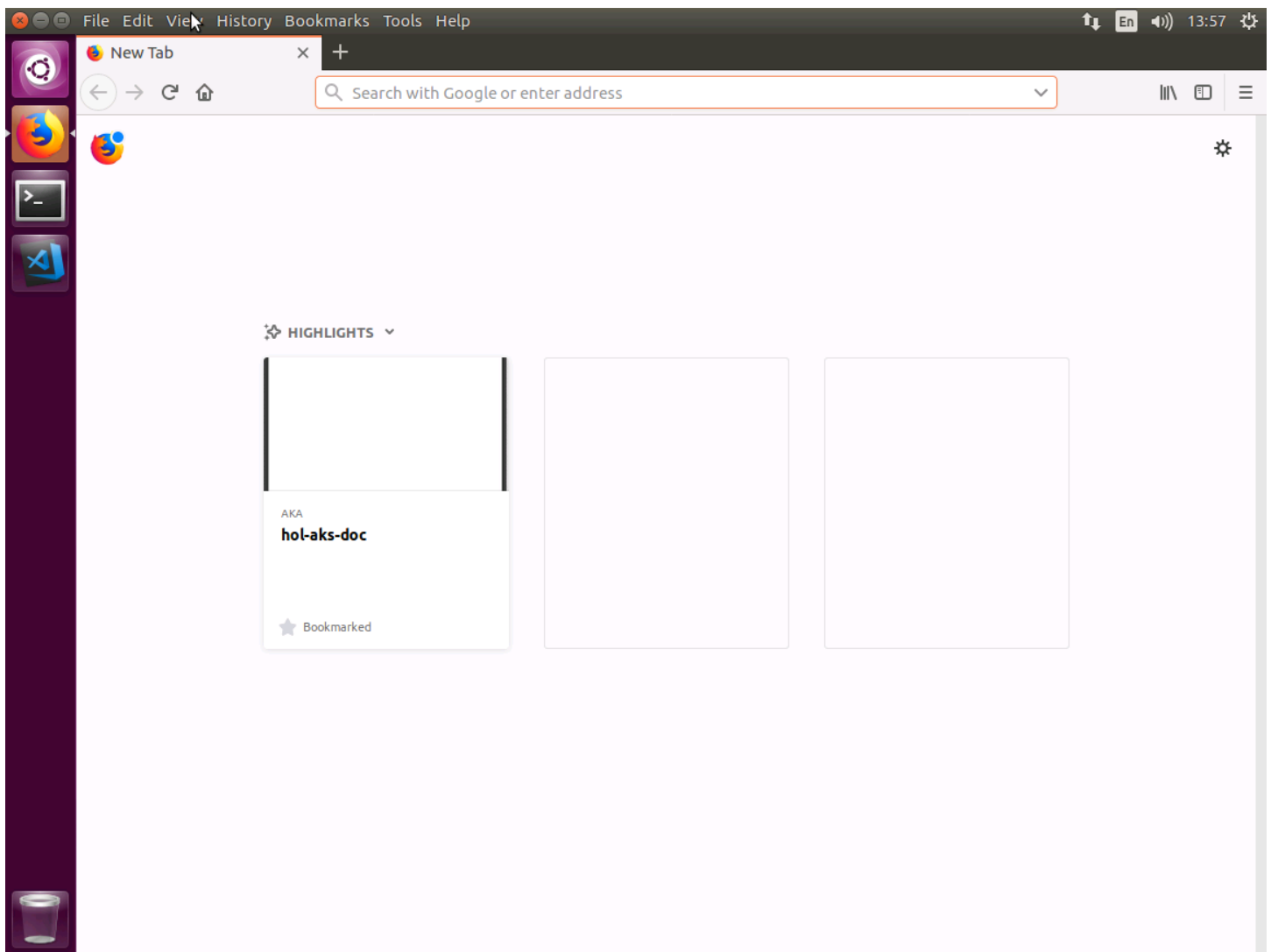
本课程为用户提供的机器中已经预先安装了所需的环境，包括

- Azure CLI
- kubectl
- Docker
- VSCODE, 并配置 kubernetes 插件

1.3. 在虚拟机中访问本文档

为了方便对文档中的命令进行复制操作，您可以在虚拟机中访问 aka.ms/hol2018-azaks 网址来访问本文档。

在我们准备的环境中，可以选择打开浏览器，然后点击‘hol-aks-doc’ 书签(指向 aka.ms/hol2018-azaks)，之后会弹出 pdf 阅读器打开本文档。



2. 创建 Azure Kubernetes 服务 (AKS) 群集

2.1. 登陆 Azure 控制台

在控制台窗口中输入

```
az login
```

然后按提示进入 <https://microsoft.com/devicelogin> 页面输入 code 登陆，使用本实验提供的账号即可。

登陆成功以后可以看到控制台返回当前账户信息。

2.2. 创建 Kubernetes 群集

在部署 Azure Kubernetes 群集时，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，方便对一组资源进行分类管理。

下面要用到的 az 命令为 Azure 2.0 版本的命令行工具。可以在 <http://aka.ms/azure-cli> 找到(本实验已经提供的虚拟机中已经安装)。

如果在实验环境中，只能使用已创建好的指定资源组，请将 RESOURCE_GROUP_NAME 设置成指定资源组并跳过创建资源组的步骤。具体情况见实际实验说明。

```
# 定义资源组名，为了防止重名，请把" group-1" 修改成指定资源组名
RESOURCE_GROUP_NAME=group-1
```

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 `group-1` 的资源组。

首先设置变量，以便在后续步骤中使用：

```
# 定义资源组名，为了防止重名，请把" group-1" 修改成您自定义的名称
RESOURCE_GROUP_NAME=group-1
LOCATION=eastus
# 创建资源组
az group create --name $RESOURCE_GROUP_NAME --location $LOCATION
```

下面的示例在 `group-1` 资源组中创建 `myAKSCluster` 群集。

```
# 制定集群名
CLUSTER_NAME=myAKSCluster001
# 创建群集
az aks create --resource-group $RESOURCE_GROUP_NAME --name $CLUSTER_NAME --node-count 1 \
  --generate-ssh-keys -k 1.11.2 --disable-rbac
```

创建大约需要 10 分钟时间，在此过程中您可以通过 Azure 门户查看创建的过程。

由于该过程用时较长，您也可以另外启动一个终端窗口，先进行如下步骤：

[3. 为 AKS 准备应用程序](#)

创建完成后会返回有关 AKS 部署的 JSON 格式信息。

2.3. 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
```

输出示例：

NAME	STATUS	ROLES	AGE	VERSION
aks-nodepool1-14171628-0	Ready	agent	1h	v1.11.2

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。

3. 为 AKS 准备应用程序

这里将准备一个要在 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

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。

3.1. 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。

3.2. 创建容器映像

```
pushd k8s-demo/nginx-server/deployment/nginx-server-image
docker build --network=host --no-cache -t nginx-hol:1.0.0 .
popd
```

完成后，使用 [docker images](#) 命令查看创建的映像。

```
docker images
```

输出示例：

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

3.3. 在本地测试应用程序

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

输出示例：

```
<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>
<a href="gitclone.sh">gitclone.sh</a>
</pre><hr></body>
</html>
```

01-May-2018 13:33	545
03-May-2018 07:26	281

3.4. 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。

```
docker rm -f nginx
```

4. 创建容器注册表

3.1 部署 Azure 容器注册表

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。

```
# 定义容器注册表名，为了防止重名，请把"acaname01"修改成您自定义的名称
ACR_NAME=acaname01
# 创建容器注册表
az acr create --resource-group $RESOURCE_GROUP_NAME --name $ACR_NAME --sku Basic
```

3.2 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。

3.3 标记容器映像

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 域名。

```
ACR_SERVER=$(az acr show -g $RESOURCE_GROUP_NAME -n $ACR_NAME --query "{.loginServer}" -o tsv)
echo $ACR_SERVER
```

输出示例：

```
acrname01.azurecr.io
```

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

```
docker tag nginx-hol:1.0.0 $ACR_SERVER/nginx-hol:1.0.0
```

标记后即可运行 `docker images` 验证操作。

```
docker images
```

输出示例：

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

3.4 将映像推送到注册表

将 `nginx-hol` 映像推送到注册表。

```
docker push $ACR_SERVER/nginx-hol:1.0.0
```

此操作需要几分钟时间完成。

3.5 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
```

输出示例：

```
Result
-----
nginx-hol
```

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-hol --output table
```

输出示例：

```
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。

5. 运行应用程序

在进行本章之前，请确认如下章节步骤已完成：

2. 创建 Azure Kubernetes 服务 (AKS) 群集
3. 为 AKS 准备应用程序
4. 创建容器注册表

5.1 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 AKS 群集有权限从 ACR 注册表拉取映像。

在之前一步创建 Kubernetes 群集中，已经自动生成了一个 Azure 应用程序，通过如下命令获取其 ID 值：

```
APP_ID=$(az aks show --resource-group $RESOURCE_GROUP_NAME --name $CLUSTER_NAME \
--query "servicePrincipalProfile.clientId" --output tsv)
echo $APP_ID
```

输出示例

```
e76c717b-396c-45ec-bea4-9b1f8a523d16
```

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" -o tsv)
echo $ACR_ID
```

输出示例

```
/subscriptions/{subs-id}/resourceGroups/grp1/providers/Microsoft.ContainerRegistry/registries/<acrName>
```

创建角色分配，以便 AKS 群集有权限从 ACR 中拉取镜像。

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

5.2 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。

首先确定我们刚刚上传的应用程序的镜像名

命令：

```
echo $ACR_SERVER/nginx-hol:1.0.0
```

示例输出：

```
acname01.azurecr.io/nginx-hol:1.0.0
```

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。

```
gedit k8s-demo/nginx-server/nginx-server-azurefile.yaml
```

将 `image:` 后的内容替换为之前上传到 Azure 容器注册表 (ACR) 的镜像名。

```
...
containers:s
- name: nginx-server
  image: acname01.azurecr.io/nginx-hol:1.0.0
```

保存并关闭该文件。

5.3 部署应用程序

使用 `kubect` [kubectl create](#) 命令部署该应用程序。此命令分析清单文件并创建定义的 Kubernetes 对象。

```
kubectl create -f k8s-demo/nginx-server/nginx-server-azurefile.yaml
```

输出示例：

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

5.4 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 40.113.236.142 80:30676/TCP 2m
```

使用如下命令查看并浏览该地址

```
EXTERNAL_IP=$(kubectl get service nginx-server -o yaml |grep ip |awk '{print $NF}')
echo $EXTERNAL_IP
sensible-browser $EXTERNAL_IP &
```

若要查看应用程序，请浏览到外部 IP 地址。

<div> <div>← → ↻</div> <div>40.113.236.142</div> </div>		
Index of /		
<hr/>		
../		
azurefile/	19-Jun-2018 09:02	-
genfiles.sh	01-May-2018 13:33	545
gitclone.sh	03-May-2018 07:26	281

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 pods
```

如果之前没有完成步骤 3，也可以直接使用 docker hub 镜像：

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

5.5 （可选步骤）测试 Azure File 功能

之前的步骤里我们的搭建了一个 nginx 应用，该应用可以浏览的文件是一个 AzureFile 的共享文件目录。

AzureFile 是 Azure 提供的基于 SMB 协议的远程文件系统，与 windows 的共享文件夹使用同一种协议。

AKS 默认会建立一个名为 MC_\${RESOURCE_GROUP_NAME}_\${CLUSTER_NAME}_\${LOCATION} 的资源组用于创建计算资源（虚拟机），和存储资源。相应的 AzureFile 的存储账户也会在这个资源组中创建。

执行如下命令在 portal 中打开该 AzureFile 资源

```
MC_GROUP="MC_${RESOURCE_GROUP_NAME}_${CLUSTER_NAME}_${LOCATION}"
echo $MC_GROUP
STORAGE_ACCOUNT_ID=$(az storage account list -g $MC_GROUP --query "[].{id:id}" -o tsv)
echo $STORAGE_ACCOUNT_ID

PORTAL_URL="https://portal.azure.com/#@/resource$STORAGE_ACCOUNT_ID/fileList"
sensible-browser $PORTAL_URL &
```

在浏览器中我们可以看到已经有一个名为 “kubernetes-dynamic-pvc-*” 的 FileShare 被创建：

Home > fa4de5b31b72b11e8905172 - Files

fa4de5b31b72b11e8905172 - Files
Storage account

Search (Ctrl+/)

Overview
Activity log
Access control (IAM)
Tags
Diagnose and solve problems
Events
Storage Explorer (preview)

Settings
Access keys
CORS
Configuration
Encryption
Shared access signature

+ File share Refresh

Storage account: fa4de5b31b72b11e8905172

Search file shares by prefix

NAME	MODIFIED	QUOTA	
kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c...	9/13/2018 4:04:50 PM	10 GiB	...

点击该 FileShare，然后点击 nginx-server，找到 README.md 文件，点击右边的“...”，然后选择“编辑”

Home > fa4de5b31b72b11e8905172 - Files > kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c2c97e507a

kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c2c97e507a
File share

Search (Ctrl+/)

Overview

Settings
Access policy
Properties

Upload Add directory Refresh Delete directory Properties

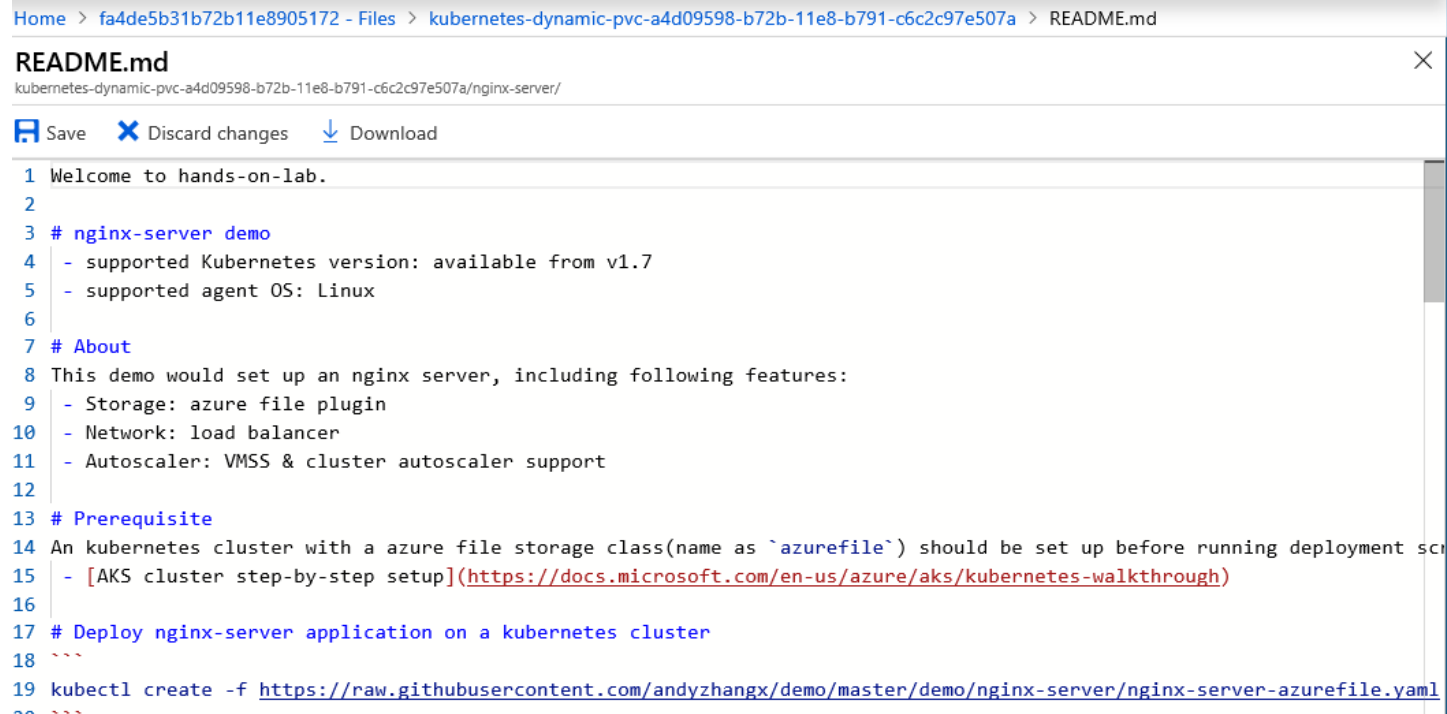
Location: kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c2c97e507a/nginx-server

Search files by prefix

NAME	TYPE	SIZE	
[..]			...
deployment	Directory		...
2018LC3-hands-on-lab-noacr.md	File		...
2018LC3-hands-on-lab.md	File		...
build-deploy-app-on-aks.pdf	File		...
nginx-server-azurefile-mooncake.yaml	File		...
nginx-server-azurefile.yaml	File		...
README.md	File	2.75 KiB	...

Edit
Download
Properties
Edit metadata
Delete

在编辑框中输入一些字符串如 “Welcome to hands-on-lab” ,然后点击保存



The screenshot shows a web-based file editor interface. At the top, there is a breadcrumb navigation: Home > fa4de5b31b72b11e8905172 - Files > kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c2c97e507a > README.md. Below this is the file name 'README.md' and its path 'kubernetes-dynamic-pvc-a4d09598-b72b-11e8-b791-c6c2c97e507a/nginx-server/'. A toolbar contains 'Save' (with a floppy disk icon), 'Discard changes' (with an 'X' icon), and 'Download' (with a download icon). The main area displays the content of the README.md file, with line numbers 1 through 19 on the left. The content includes a welcome message, a section for 'nginx-server demo' with supported Kubernetes versions and OS, an 'About' section, a 'Prerequisite' section, and deployment instructions.

```
1 Welcome to hands-on-lab.
2
3 # nginx-server demo
4 - supported Kubernetes version: available from v1.7
5 - supported agent OS: Linux
6
7 # About
8 This demo would set up an nginx server, including following features:
9 - Storage: azure file plugin
10 - Network: load balancer
11 - Autoscaler: VMSS & cluster autoscaler support
12
13 # Prerequisite
14 An kubernetes cluster with a azure file storage class(name as `azurefile`) should be set up before running deployment script
15 - [AKS cluster step-by-step setup](https://docs.microsoft.com/en-us/azure/aks/kubernetes-walkthrough)
16
17 # Deploy nginx-server application on a kubernetes cluster
18 ```
19 kubectl create -f https://raw.githubusercontent.com/andyzhangx/demo/master/demo/nginx-server/nginx-server-azurefile.yaml
20 ```
```

使用如下命令查看修改的结果：

```
FILE_URL=http://$EXTERNAL_IP/azurefile/nginx-server/README.md
curl -s $FILE_URL | head
```

输出示例：

```
Welcome to hands-on-lab

# nginx-server demo
- supported Kubernetes version: available from v1.7
- supported agent OS: Linux
```

6 缩放应用程序

6.1 手动缩放 Deployment

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
```

输出示例：

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
```

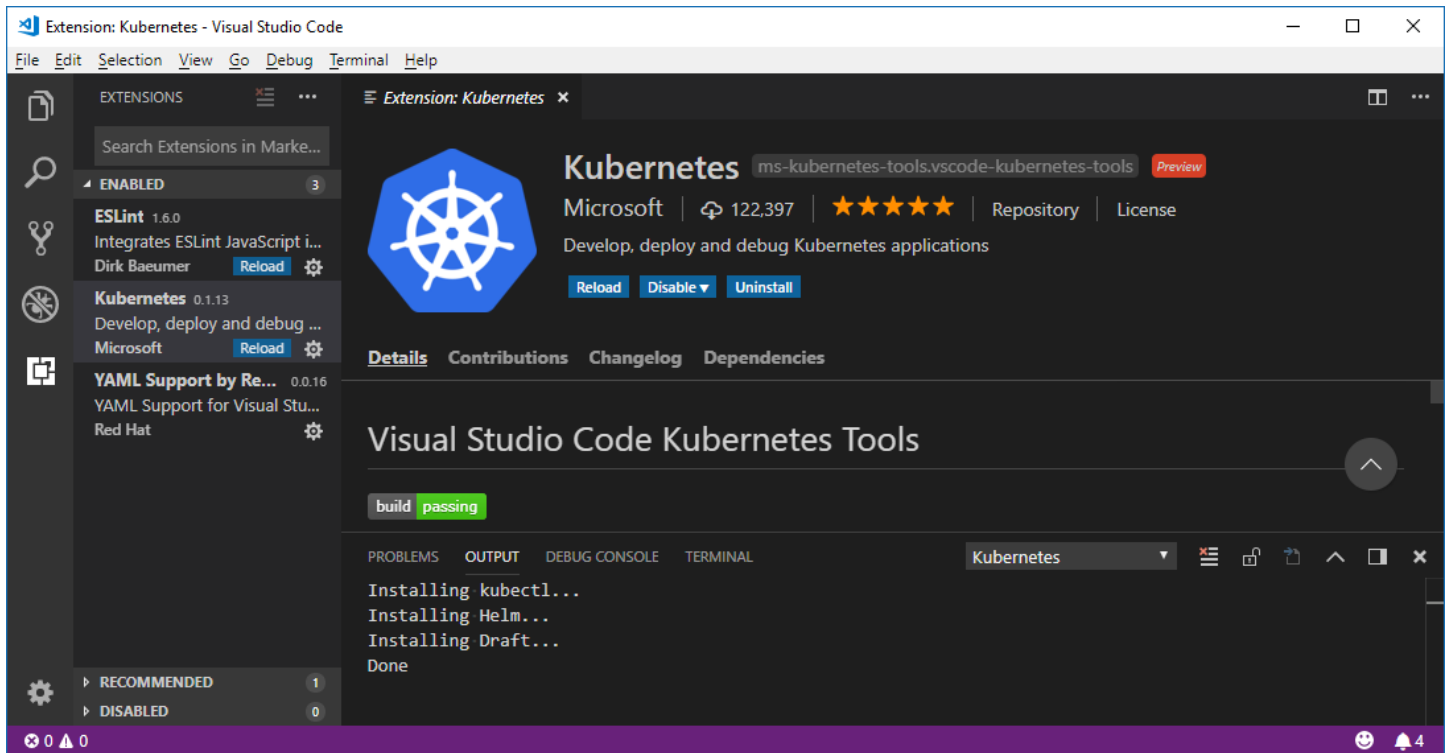
输出示例：

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

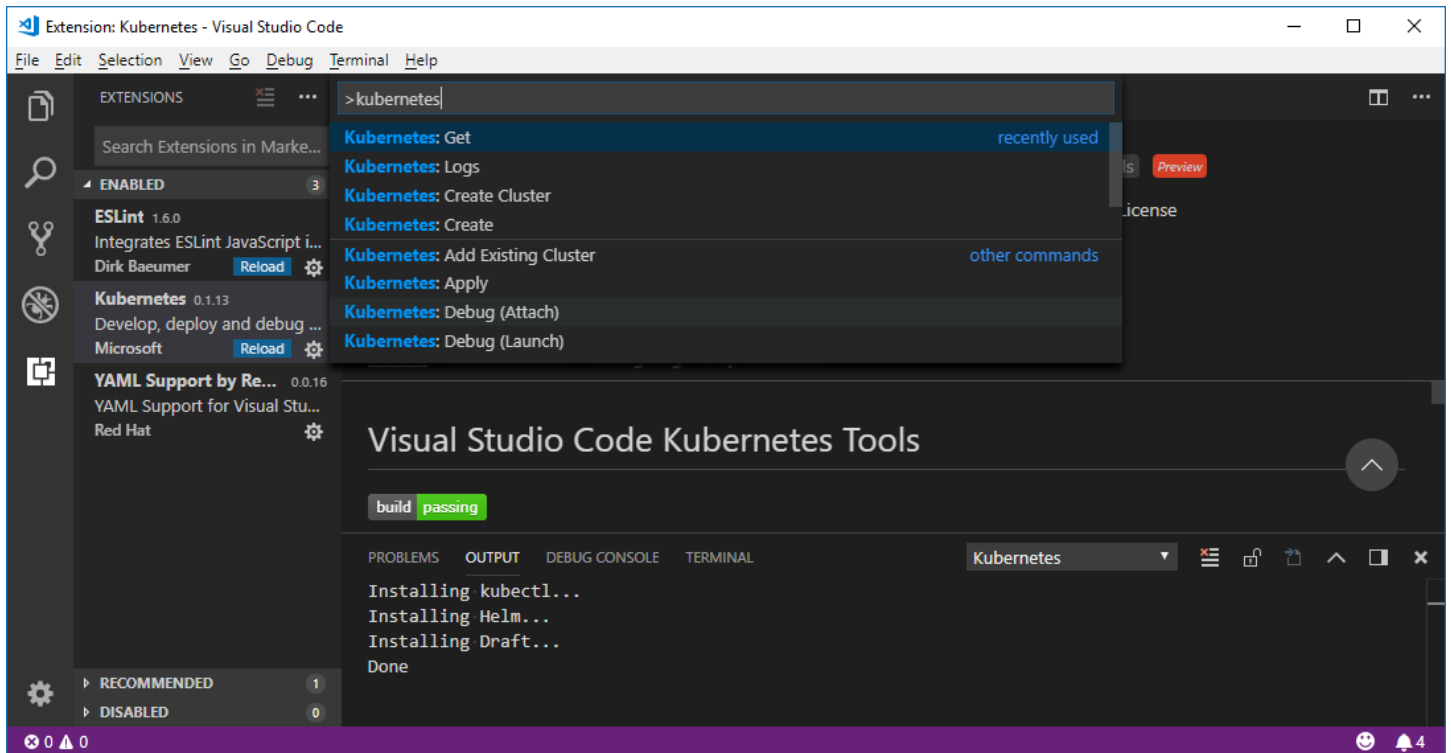
6.2 通过 VSCODE 扩展控制缩放 Deployment

同样地，我们可以通过 Visual Studio Code (VSCODE) 的扩展来操作 kubernetes。该扩展提供开发过程中一种操作 kubernetes 的便利方式。

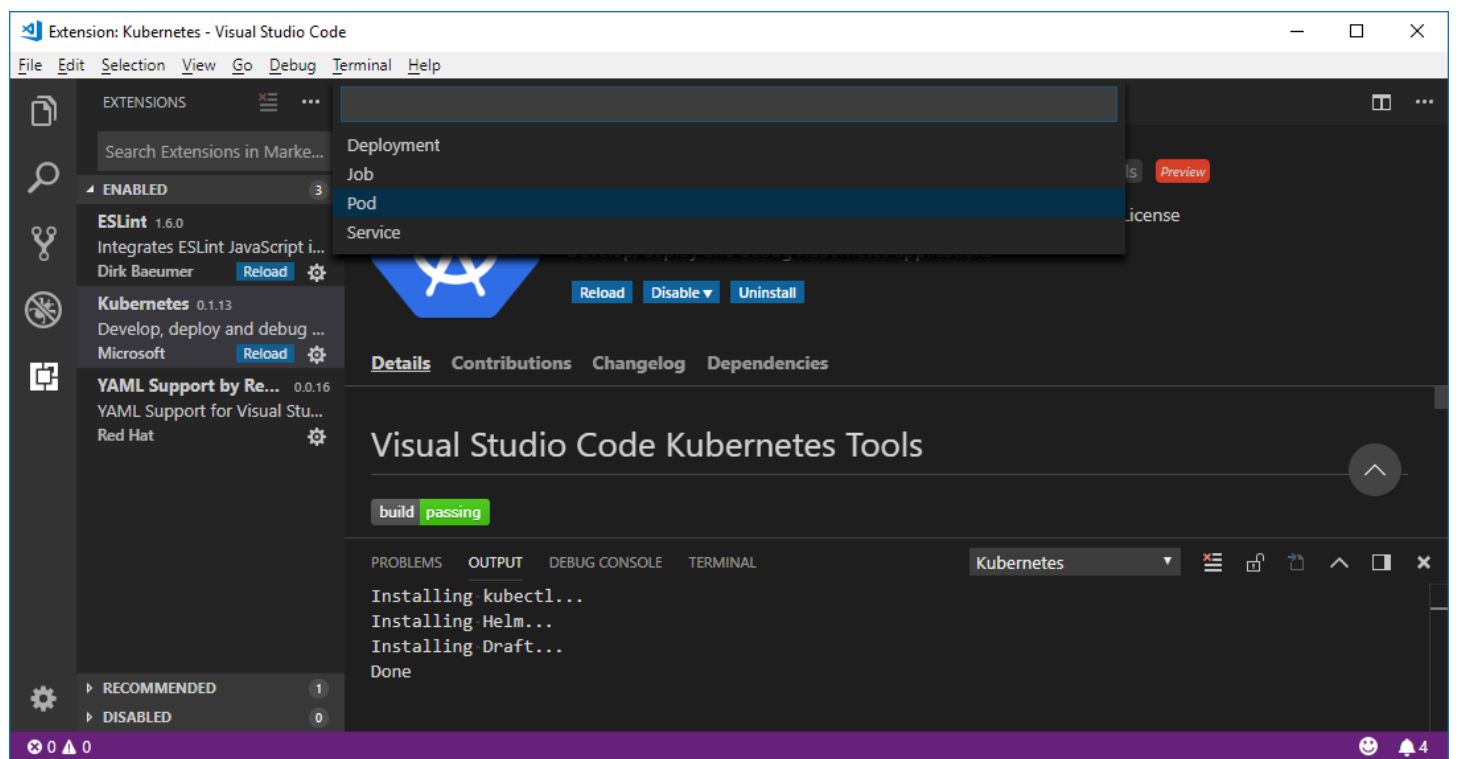
我们首先需要安装 VSCODE 的 kubernetes 扩展，打开扩展面板，搜索并确认 kubernetes 已安装。



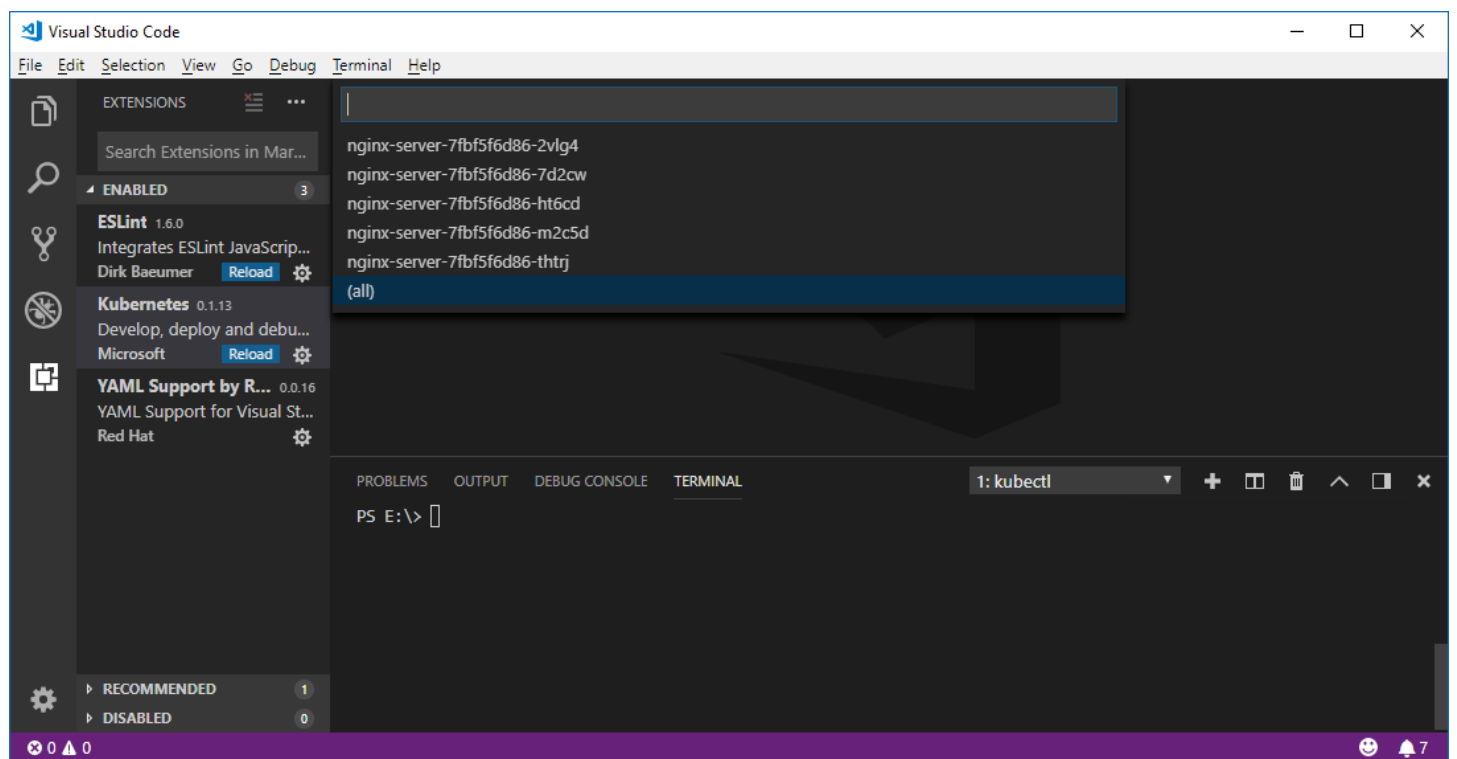
安装好以后现在我们按下 F1 并输入 kubernetes，可以看到一系列 VSCODE 支持的命令。



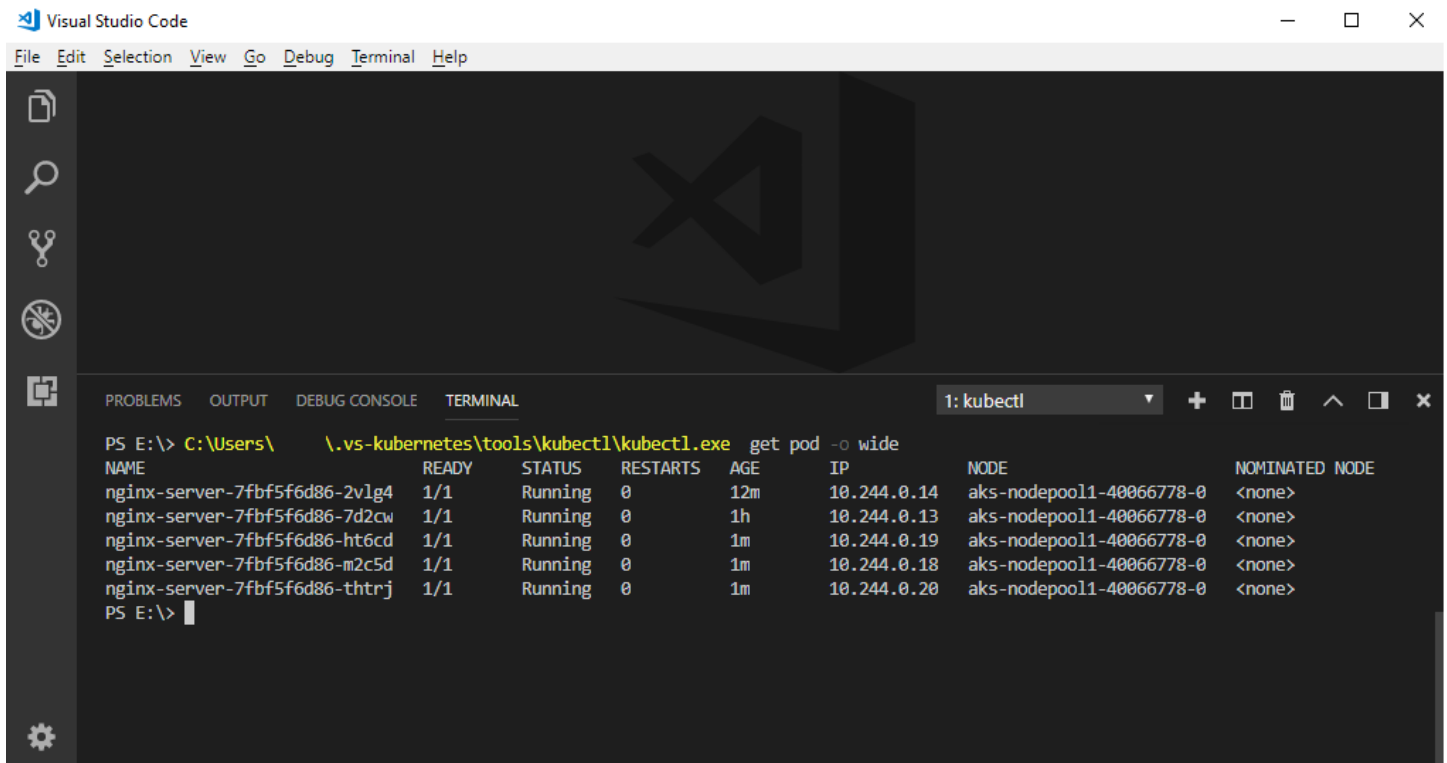
这里我们选择“Kubernetes: Get”，出现 “What resource do you want to get?” 提示时按回车，跳转到如下选择界面。这里选择 pod，表示我们要列出当前运行的 pod 状态



接下来界面会列出现有的 pods，我们选择(all)

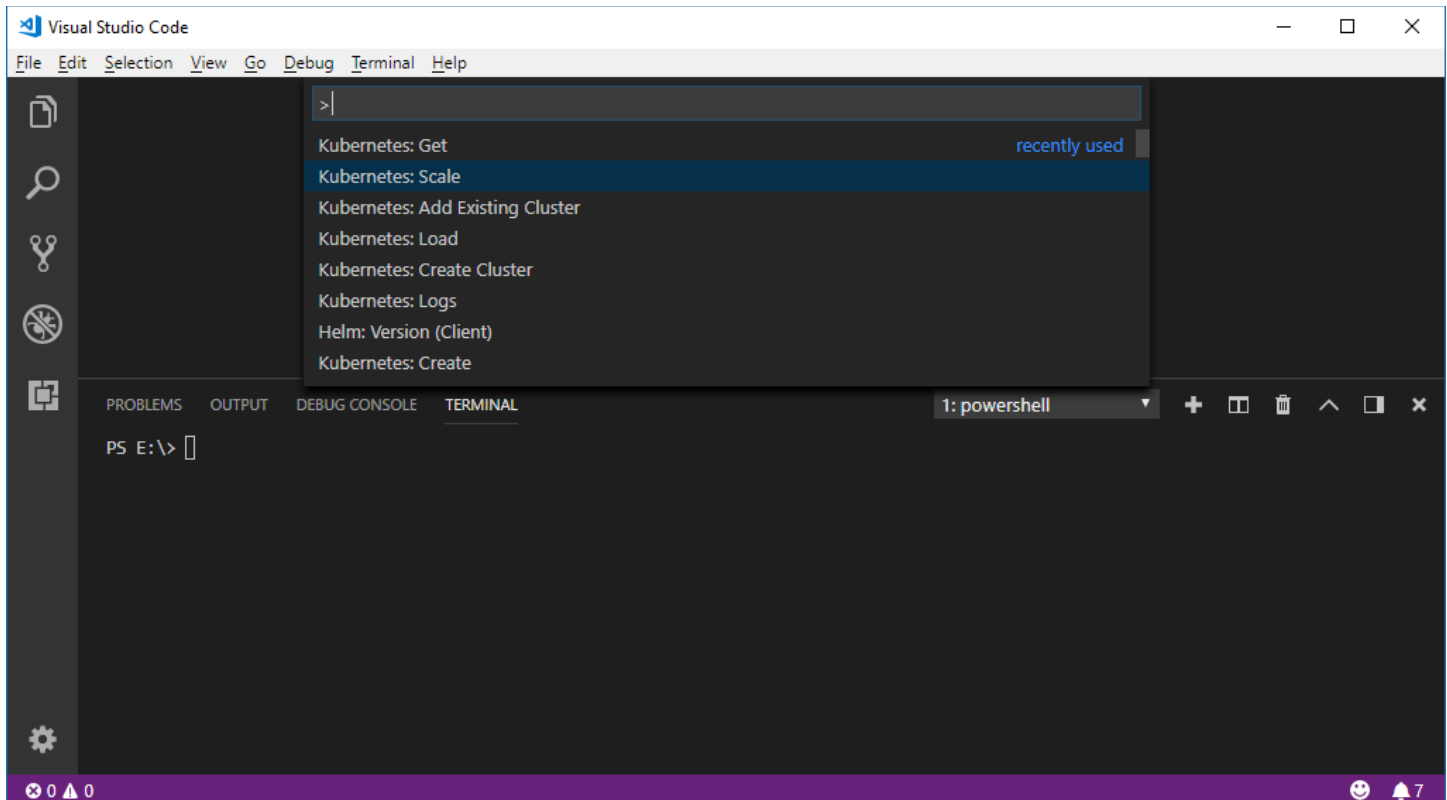


可以看到在 TERMINAL 窗口中显示了执行的命令，以及我们现在有 5 个 pod 在运行。



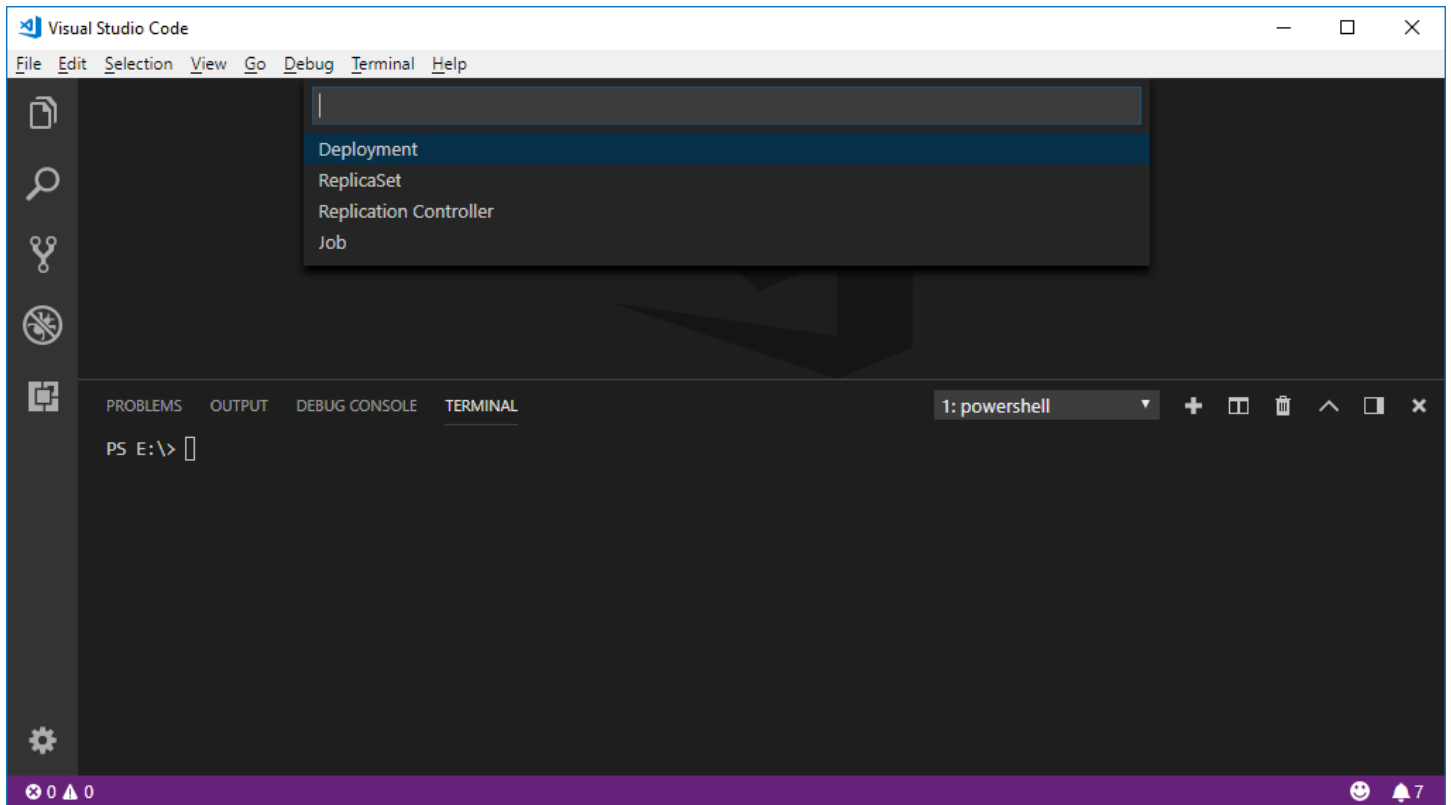
```
PS E:\> C:\Users\ \.vs-kubernetes\tools\kubectl\kubectl.exe get pod -o wide
NAME                                READY    STATUS    RESTARTS   AGE    IP             NODE                                NOMINATED NODE
nginx-server-7fbf5f6d86-2vlg4        1/1      Running   0           12m    10.244.0.14    aks-nodepool1-40066778-0           <none>
nginx-server-7fbf5f6d86-7d2cw        1/1      Running   0           1h     10.244.0.13    aks-nodepool1-40066778-0           <none>
nginx-server-7fbf5f6d86-ht6cd        1/1      Running   0           1m     10.244.0.19    aks-nodepool1-40066778-0           <none>
nginx-server-7fbf5f6d86-m2c5d        1/1      Running   0           1m     10.244.0.18    aks-nodepool1-40066778-0           <none>
nginx-server-7fbf5f6d86-thtrj        1/1      Running   0           1m     10.244.0.20    aks-nodepool1-40066778-0           <none>
PS E:\>
```

再次按下 F1, 输入 kubernetes , 然后选择 scale。

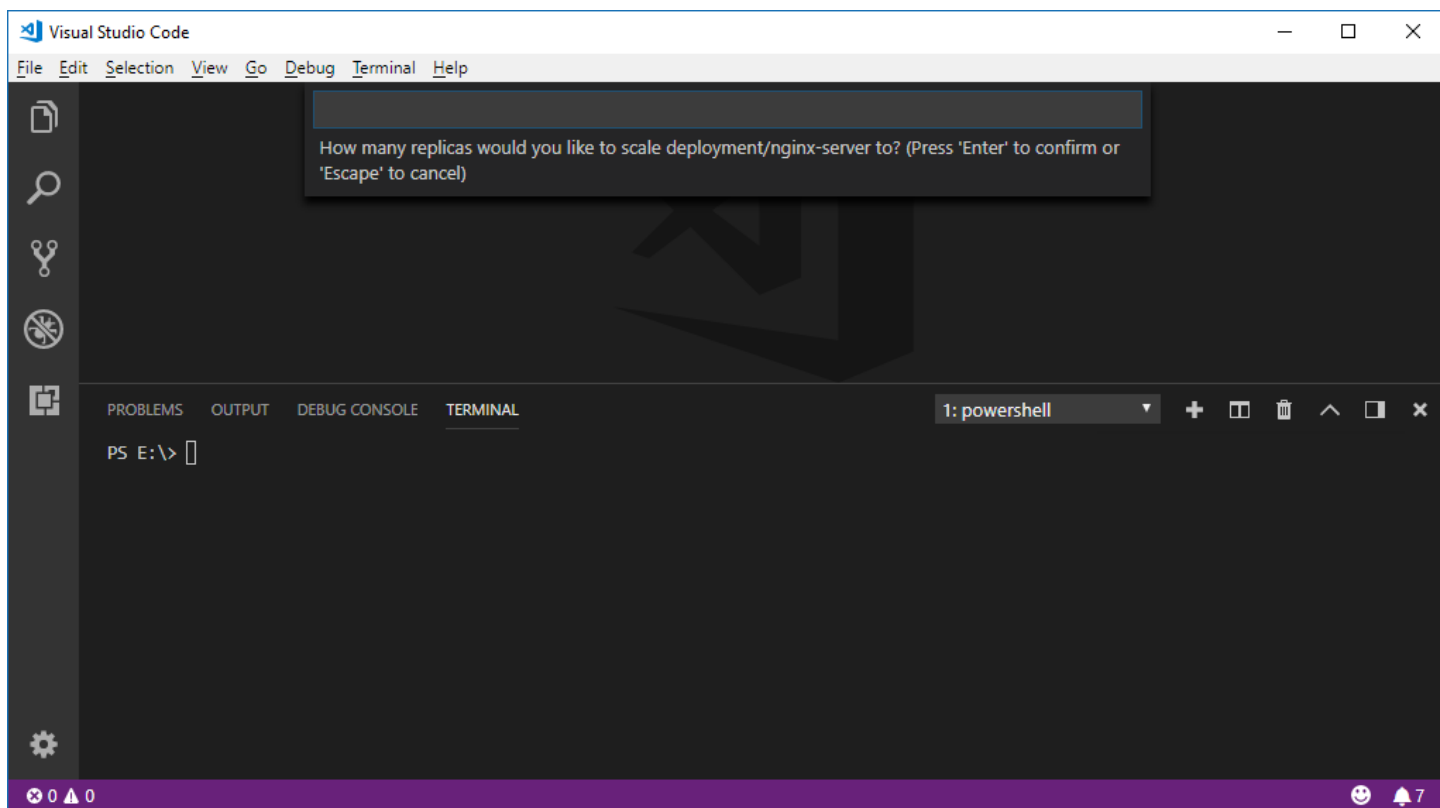


```
>|
Kubernetes: Get
Kubernetes: Scale
Kubernetes: Add Existing Cluster
Kubernetes: Load
Kubernetes: Create Cluster
Kubernetes: Logs
Helm: Version (Client)
Kubernetes: Create
```

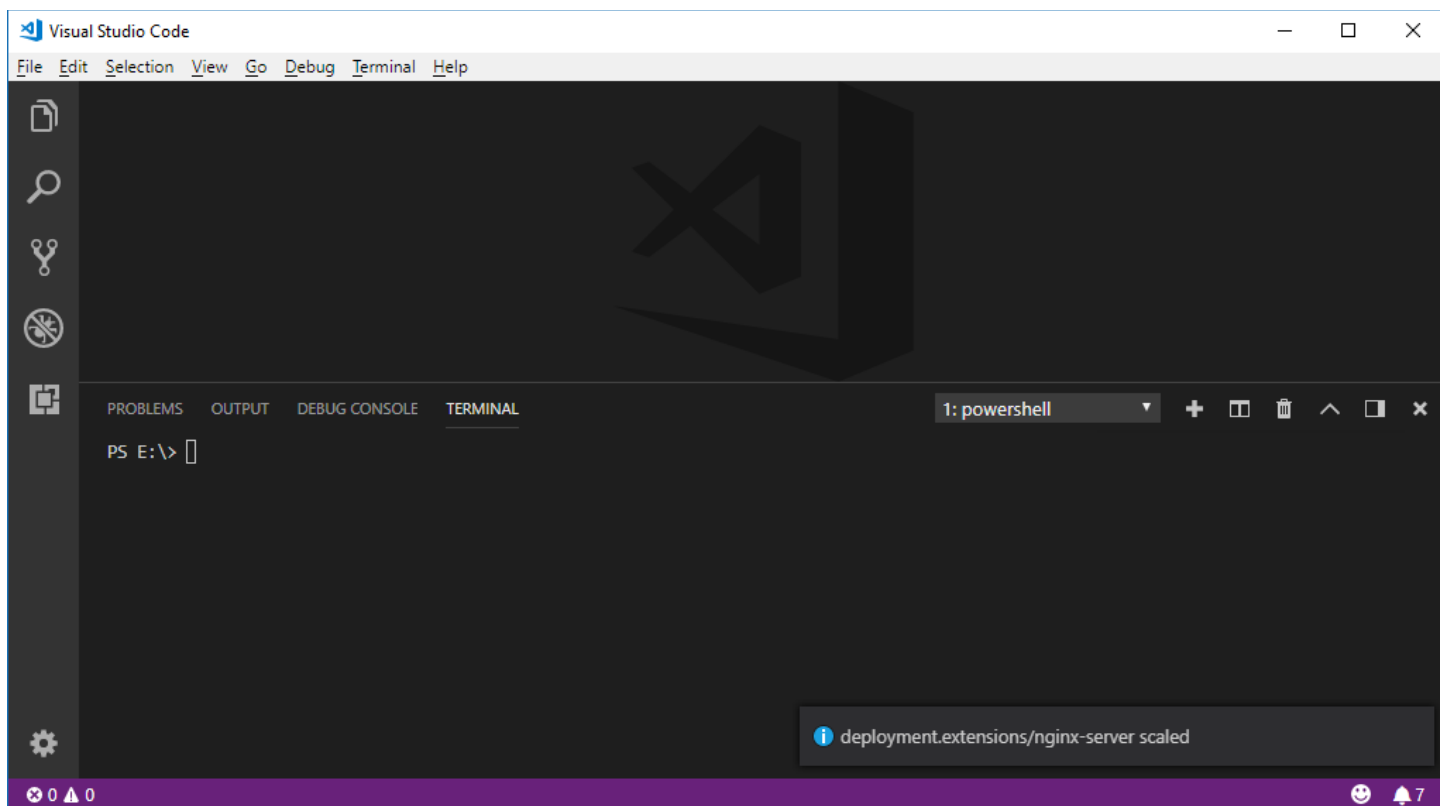
出现 “What resource do you want to get?” 提示时按回车，跳到如下选择界面。这里选择 Deployment 表示对 deployment 进行操作。



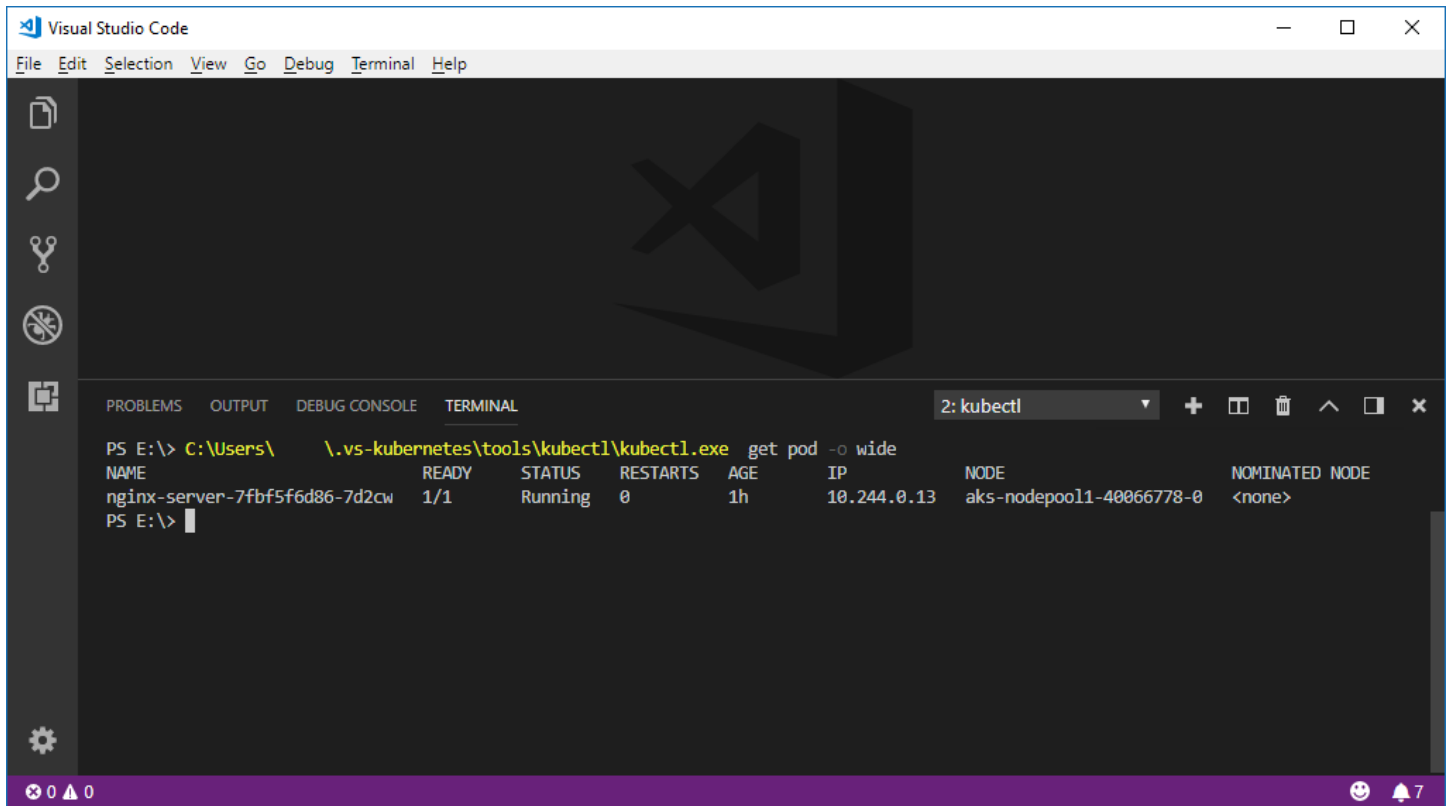
这时候我们部署的 deployment “nginx-server” 被列出，直接选择后出现如下界面。在这里我们输入目标数量副本 1。



可以看到提示 scale 执行完成。



现在我们重复前面 get pod 的操作，列出现在 pod 状态。



The screenshot shows a Visual Studio Code window with a terminal open. The terminal is running the command `kubectl get pod -o wide`. The output is a table with columns: NAME, READY, STATUS, RESTARTS, AGE, IP, NODE, and Nominated Node. The table shows one pod named `nginx-server-7fbf5f6d86-7d2cw` in a `READY` state, running on node `aks-nodepool1-40066778-0`.

NAME	READY	STATUS	RESTARTS	AGE	IP	NODE	Nominated Node
nginx-server-7fbf5f6d86-7d2cw	1/1	Running	0	1h	10.244.0.13	aks-nodepool1-40066778-0	<none>

可以看到，pod 的数量又恢复成 1 个。相当与执行了“`kubectl scale --replicas=1 deployment/nginx-server`”命令。

6.3 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
```

输出示例：

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

到这里本教程的内容已经完成，您可以参考 <https://github.com/kubernetes/examples>，在 Azure 部署更多的 Kubernetes 应用案例。

感谢您的参与。

1. 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)