Microsoft Azure Administrator Associate Training

Azure app services, Web App for Containers, WebJobs & Azure Kubernetes Services

Agenda

- ❑ What is Azure Apps Service
- ❑ What is Azure Mobile App
- ❑ What is Azure Logic Apps
- ❑ Deploying Web Apps
- ❑ Comparison of VM vs WebApp vs Cloud Services
- ❑ FTP Transfer
- ❑ App Service Web App for Containers
- ❑ WebJobs
- ❑ Diagnostics Logging
- ❑ Azure Kubernetes Service

Azure Apps Service

What Is Azure Apps Service?

- ❑ App Service provides a comprehensive platform for building cloud-based applications.
- ❑ App Service provides a hosted service with which developers can build mobile and web apps.
- ❑ The Web Apps feature is a platform of technologies that enable you to build web apps in Azure without having to deploy, configure, and maintain your own Azure VMs.
- ❑ You can build web apps by using the ASP.NET, PHP, Node.js, Java, and Python frameworks and a range of programming languages, such as C#, HTML5, PHP, Java, Node.js, and Python.
- ❑ They also integrate with common development environments such as Visual Studio and GitHub.
- ❑ With App Service, you pay for the Azure compute resources you use.

WEB APP
MOBILE APP
API APP
LOGIC APP

Azure Web Apps: Features

Visual Studio integration

- Dedicated tools in Visual Studio streamline the work of creating, deploying, and debugging.

API and mobile features

- Web Apps provides turn-key CORS support for RESTful API scenarios and simplifies mobile app scenarios by enabling authentication, offline data sync, push notifications, and more.

Serverless code

- Run a code snippet or script on-demand without having to explicitly provision or manage infrastructure, and pay only for the compute time your code actually uses.

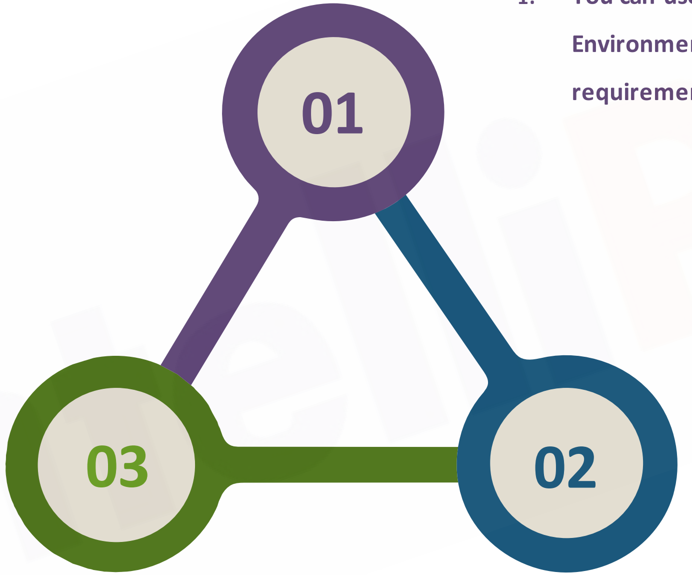
What Is Azure Mobile App?

- ❑ The Mobile Apps feature is a part of App Service.
- ❑ It provides a platform for building and hosting backend services for mobile applications.
- ❑ The Mobile Apps feature allows developers to build cross-platform apps that can run on Windows, iOS, or Android.
- ❑ These apps can operate exclusively in the cloud or connect with your on-premises infrastructure for authentication and authorization purposes.
- ❑ They can benefit from the built-in push notification engine that can send personalized push notifications to almost any mobile device that is using iOS, Android, or Windows.

What Is Azure Logic Apps?

1. Logic apps automate business processes by linking together cloud-based apps, such as Office 365, Google Services, and Salesforce.
2. With the Logic Apps feature, you can use a visual designer to combine connectors available from Azure Marketplace for different integration scenarios.
3. Logic Apps uses a workflow engine to implement business processes that you designed and relies on connectors to provide user access.
4. Connectors also facilitate initiating new workflow instances via triggers based on events that you define.
5. Each step in the workflow is an action that accesses data or services through a connector.
6. More advanced integration scenarios can use rules, transformations, validations, and features that are part of BizTalk Services.

Azure App Service Environment



1. You can use the App Service Environment to accommodate this requirement.

2. Business-critical apps often require highly scalable, isolated, and dedicated environments.

3. You can use the App Service Environment to host web apps, mobile apps, and API apps that require highly scalable compute resources, isolation, and direct virtual network connectivity.

Azure Web Apps: Deploying Web Apps

- ❑ You can deploy your web apps by using several methods:

Copying files manually by using FTP

Synchronizing files and folders to App Service from a cloud storage service, such as OneDrive or Dropbox.

- ❑ App Service also supports deployments by using the Web Deploy technology.
- ❑ This approach is available with Visual Studio, WebMatrix, and Visual Studio Team Services.
- ❑ If you want to perform deployments by using Git or FTP, you must configure deployment credentials.
- ❑ Knowledge of deployment credentials will allow you to upload the web app's code and content to the new web app to make it available for browsing.

Azure Web Apps: Comparison

To host a web application in Azure, you can use:

Azure VMs, Web Apps, or Azure Cloud Services.

Azure VMs

- Azure VM can host any web server, such as IIS or Apache.
- You can host supporting servers, such as SQL Server instances that host databases, on other VMs.
- You can use Azure VM load balancing or Azure VM Scale Sets to scale the web application horizontally.
- If you choose to host a web application on Azure VMs, you have maximum control over their OS and supporting software components.

Web Apps

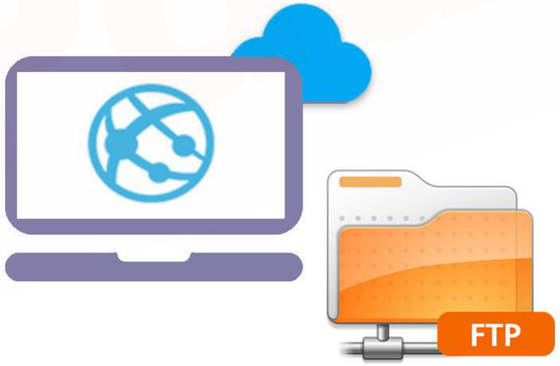
- Alternatively, you can choose to host your web application using the Web Apps feature.
- You can upload a custom web application code into it or deploy any code from the Azure.
- You can also scale web apps horizontally by changing the number of instances and relying on Azure built-in load balancing to distribute the traffic across them.
- You also cannot establish an RDP or SSH connection to the virtual machine hosting a web app.

Cloud Services

- You also can choose to build a web application as a cloud service.
- A cloud service consists of a web role, which serves as the front-end of an application, and a worker role, which is responsible for running background tasks.
- You can scale each role independently by specifying the number of role instances, which gives you more control over scalability compared to web apps.
- Since role instances run Windows Server, you can connect to them using RDP.
- PaaS cloud services are unique to Azure.
- Existing web applications might require significant modification before they can run as a cloud service.

Azure Web Apps: FTP Transfer

- To deploy a web app by using FTP, you must configure your client with the destination URL of the remote FTP server and the credentials that FTP can use to authenticate.
- In addition, you must choose either the active or the passive FTP mode.
- These are the Azure web app deployment credentials.



Hands-on

Hands-on

- ❑ Configure and deploy the code via Web App
- ❑ Configure and deploy the code via the cloud storage service

App Service Web App for Containers

Web App for Containers

Now, you can easily deploy and run the container-based application on your Windows and Linux which scale according to your business.

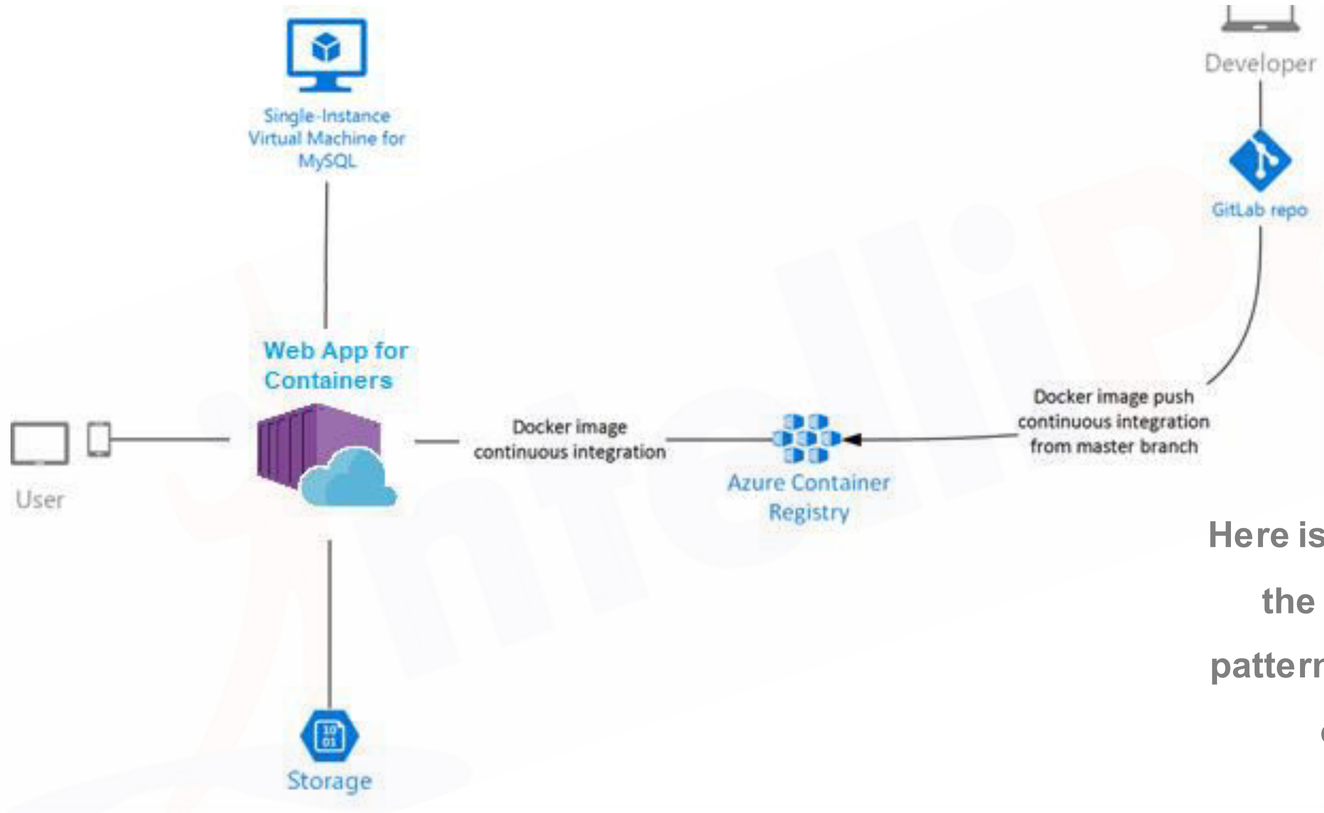
Benefits of Using Web App for Containers

- A fully managed platform to perform infrastructure maintenance
- Built-in auto scaling and load balancing
- Easily streamline CI/CD with Docker Hub, Azure Container Registry, and GitHub

Web App for Containers

- Web App for containers is catered more towards developers who want to have more control over, not just the code, but also the different packages, runtime framework, tooling, etc. that are installed on their containers.
- This offering is essentially bringing years worth of Azure App Service PaaS innovations to the community, by allowing developers to just focus on composing their containers without worrying about managing and maintaining an underlying container orchestrator. Customers of this offering prefer to package their code and dependencies into containers using various CI/CD systems like Jenkins, Maven, Travis CI, or Azure DevOps, alongside setting up continuous deployment webhooks with App Service.

Web App for Containers



Developer
GitLab repo
Single-Instance Virtual Machine for MySQL
Web App for Containers
Docker image push continuous integration from master branch
Docker image continuous integration
Azure Container Registry
User
Storage

Here is an example of one of the many architecture patterns used by customers of this offering.

Deploying a Web App for Containers

Opening Azure Cloud Shell

- Configure a deployment user, for which use the below command:

```
az webapp deployment user set --user-name <username> --password <password>
```

By this, you will get a JSON output with the password shown as null.

- Now, create a resource group using the below command:

```
az group create --name myResourceGroup --location "West Europe"
```

- You generally create your resource group and the resources in a region near to you.
- When the command finishes, a JSON output shows you the resource group properties.

Deploying a Web App for Containers

Creating an Azure App Service Plan

- In the Cloud Shell, create an App Service plan in the resource group with this command:

```
az appservice plan create --name myAppServicePlan --resource-group myResourceGroup --sku B1 --is-linux
```

- When the App Service plan has been created, the Azure CLI shows information similar to the below example:

```
{ "adminSiteName": null,
"appServicePlanName": "myAppServicePlan",
"geoRegion": "West Europe",
"hostingEnvironmentProfile": null,
"id": "/subscriptions/0000-
0000/resourceGroups/myResourceGroup/providers/Microsoft.Web/serverfarms/myAppServicePlan",
"kind": "linux",
"location": "West Europe",
"maximumNumberOfWorkers": 1,
"name": "myAppServicePlan",
< JSON data removed for brevity. >
"targetWorkerSizeId": 0,
"type": "Microsoft.Web/serverfarms",
"workerTierName": null }
```

Deploying a Web App for Containers

Creating a Web App

- Create a web app using this command:

```
az webapp create --resource-group myResourceGroup --plan myAppServicePlan --name <app name> --deployment-container-image-name
microsoft/azure-appservices-go-quickstart
```

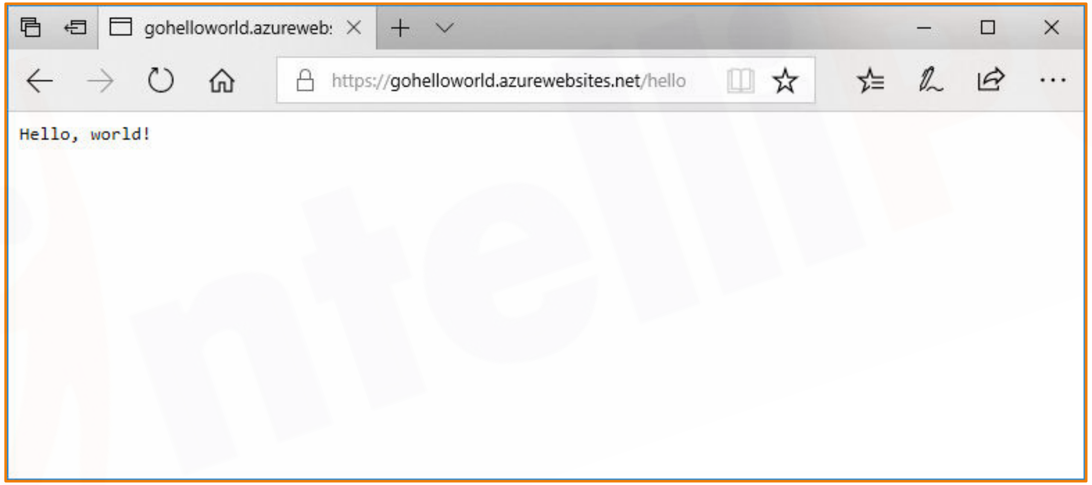
- When the web app has been created, the Azure CLI shows output similar to the below example:

```
{
"availabilityState": "Normal",
"clientAffinityEnabled": true,
"clientCertEnabled": false,
"cloningInfo": null,
"containerSize": 0,
"dailyMemoryTimeQuota": 0,
"defaultHostName": "<app name>.azurewebsites.net",
"deploymentLocalGitUrl": "https://<username>@<app
name>.scm.azurewebsites.net/<app name>.git",
"enabled": true,
< JSON data removed for brevity. >
}
```

Deploying a Web App for Containers

- Browse to the app:

http://<app_name>.azurewebsites.net/hello



```
Hello, world!
```

- Congratulations! You've deployed a custom Docker image running an application to Web App for Containers.

Hands-on

Hands-on

- ❑ Deploy a custom Docker image to Azure
- ❑ Configure environment variables to run the container
- ❑ Update the Docker image and redeploy it
- ❑ Connect to the container using SSH
- ❑ Deploy a private Docker image to Azure

Hint: In a terminal window, run the following command to clone the sample app repository to your local machine, and then change to the directory that contains the sample code.

```
>> git clone https://github.com/Azure-Samples/docker-django-webapp-linux.git --config core.autocrlf=input
>> cd docker-django-webapp-linux
```

WebJobs

WebJobs: Overview

WebJobs is a feature of Azure App Service that enables you to run a program or script in the same context as a web app, API app, or mobile app. There is no additional cost to use WebJobs.

WebJob Types

Continuous	Triggered
Starts immediately when the WebJob is created. To keep the job from ending, the program or script typically does its work inside an endless loop. If the job ends, you can restart it.	Starts only when triggered manually or on a schedule.
Runs on all instances that the web app runs on. You can optionally restrict the WebJob to a single instance.	Runs on a single instance that Azure selects for load balancing.
Supports remote debugging.	Doesn't support remote debugging.

Note: WebJobs is not yet supported for App Service on Linux.

Run Background Tasks in Azure App Service (WebJobs)

Steps to Follow

- Create a continuous WebJob
- Create a manually triggered WebJob
- Create a scheduled WebJob
- View the job history

Hands-on

Hands-on

- ❑ Develop and deploy WebJobs using Visual Studio Azure App Service Configure environment variables to run the container
  - Create and publish WebJobs as .NET Core console apps
  - Publish a .NET Core WebJob to App Service from Visual Studio
  - Enable WebJobs deployment for an existing Console Application project
  - Schedule a triggered WebJob (using CRON expression)

You may take a reference from the official Microsoft document:
https://docs.microsoft.com/en-us/azure/app-service/webjobs-dotnet-deploy-vs

Diagnostics Logging

Diagnostics Logging: Overview

With Azure diagnostic logs, you can view core analytics and save them into one or more destinations including:

- Azure Storage Account
- Azure Event Hubs
- Log Analytics Workspace

This feature is available on CDN endpoints for all pricing tiers.

How Does It Help?

Azure diagnostics logs allow you to export basic usage metrics from your CDN endpoint to a variety of sources so that you can consume them in a customized way. For example, you can do the following types of data export:

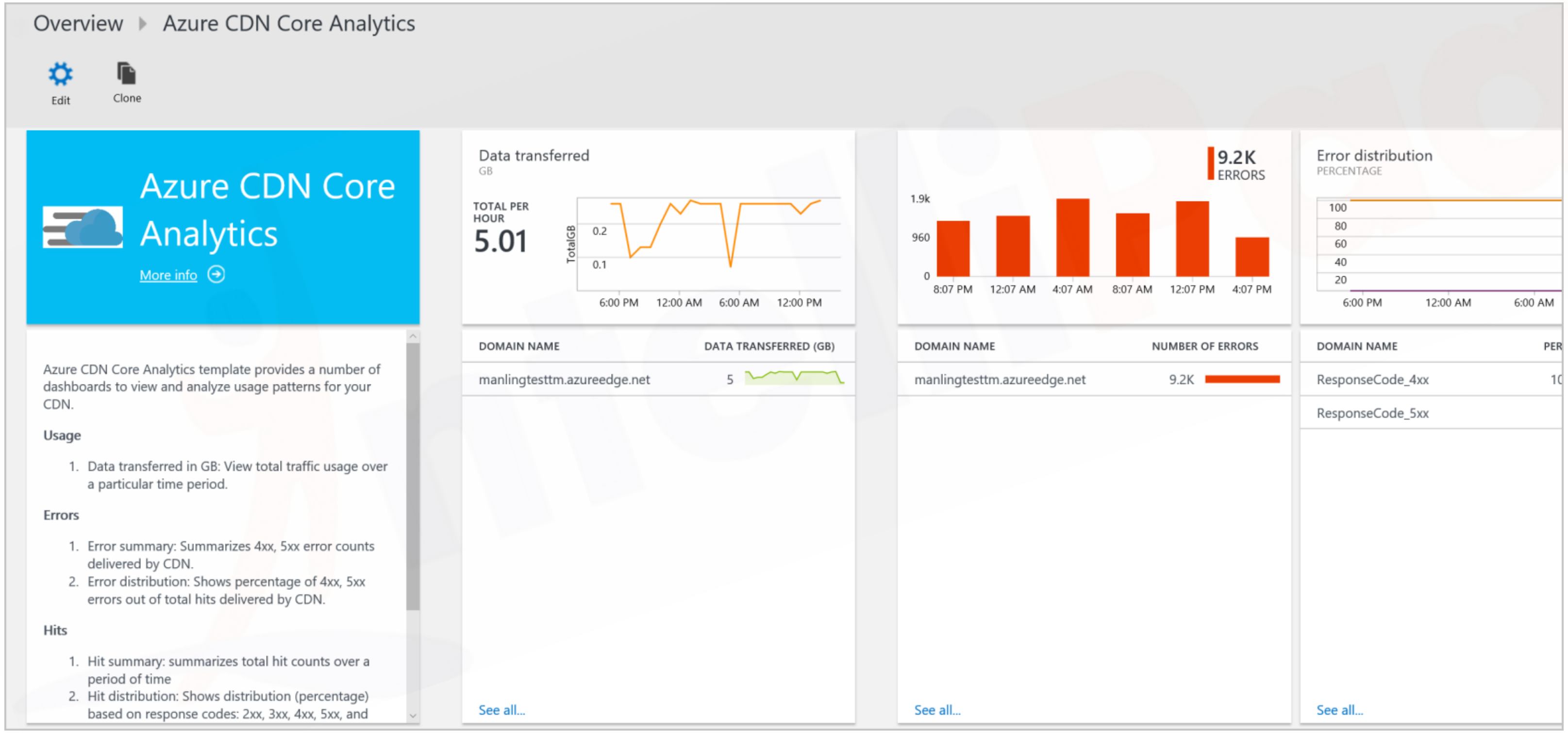
Export data to blob storage, export to CSV, and generate graphs in Excel

Export data to Event Hubs and correlate with data from other Azure services

Export data to Azure Monitor logs and view data in your own Log Analytics workspace

CDN Core Analytics View

The following diagram shows a typical CDN core analytics view of data.



Azure CDN Core Analytics template provides a number of dashboards to view and analyze usage patterns for your CDN.

Usage

1. Data transferred in GB: View total traffic usage over a particular time period.

Errors

1. Error summary: Summarizes 4xx, 5xx error counts delivered by CDN.
2. Error distribution: Shows percentage of 4xx, 5xx errors out of total hits delivered by CDN.

Hits

1. Hit summary: summarizes total hit counts over a period of time
2. Hit distribution: Shows distribution (percentage) based on response codes: 2xx, 3xx, 4xx, 5xx, and

DOMAIN NAME	DATA TRANSFERRED (GB)
manlingtesttm.azureedge.net	5

DOMAIN NAME	NUMBER OF ERRORS
manlingtesttm.azureedge.net	9.2K

DOMAIN NAME	PER
ResponseCode_4xx	10
ResponseCode_5xx	

Enabling Diagnostics Logging for Apps in Azure App Service

The first step is to do the application diagnostics.

Application diagnostics allows you to capture information produced by a web application. ASP.NET applications can use the System.Diagnostics.Trace class to log information to the application diagnostics log. For example:

```
System.Diagnostics.Trace.TraceError("If you're seeing this, something bad happened");
```

At runtime, you can retrieve these logs to help with troubleshooting.

App Service also logs deployment information when you publish content to an app. It happens automatically, and there are no configuration settings for deployment logging. Deployment logging allows you to determine why a deployment failed. For example, if you use a custom deployment script, you might use deployment logging to determine why the script is failing.

Enabling Diagnostics Logging for Apps in Azure App Service

- Steps to follow to enable the diagnostics logs
- Click on Settings > Diagnostics logs
- Choose the **Level**

For Application logging, you can turn on the file system option temporarily for debugging purposes. This option turns off automatically in 12 hours. You can also turn on the blob storage option to select a blob container to write logs to.

Next step is to **Download the Logs**

Diagnostic information stored to the app file system can be accessed directly using FTP. It can also be downloaded as a Zip archive using Azure CLI.

Enabling Diagnostics Logging for Apps in Azure App Service

- Download with **Azure CLI**
- **To download the log files using the Azure Command**
  - Line Interface
  - Open a new command prompt
  - PowerShell, Bash, or Terminal session
- **After completing the above steps, use the below command:**

```
az webapp log download --resource-group resourcegroupname --name appname
```

- This command saves the logs for the app named ***appname*** to a file named diagnostics.zip in the current directory.

Enabling Diagnostics Logging for Apps in Azure App Service

- Now, view logs in Application Insights by adding the app **SDK** in **VS** and then add the **Trace Listener package** to your project. Next, **upload** and **generate the log data**.
- Next step is to **Stream logs** with Azure CLI and **check** all your logs.

Azure Kubernetes Service

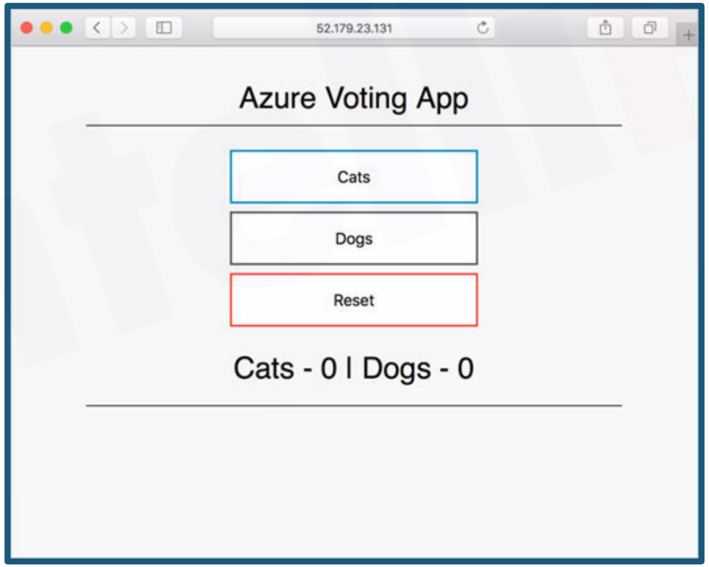
Azure Kubernetes Service

Azure Kubernetes Service (AKS) makes it simple to deploy a managed Kubernetes cluster in Azure. Basically, it reduces the complexity and operational overhead of managing Kubernetes by offloading much of that responsibility to Azure.

- As a hosted Kubernetes service, Azure handles critical tasks like health monitoring and maintenance for you.
- Kubernetes masters are managed by Azure.
- You only manage and maintain agent nodes.
- As a managed Kubernetes service, AKS is free—you only need to pay for the agent nodes within your clusters, not for the masters.

Deploying an AKS Cluster Using the Azure CLI

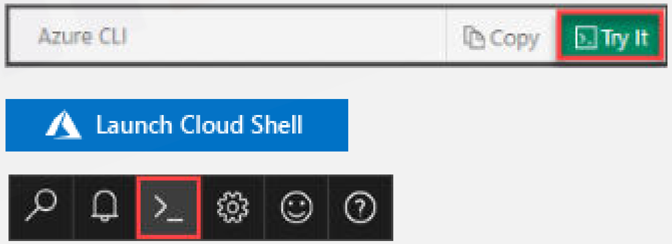
AKS is a managed Kubernetes service that lets you quickly deploy and manage clusters. In this quick start, you deploy an AKS cluster using the Azure CLI. A multi-container application that includes a web front end and a Redis instance is run in the cluster. You can then see how to monitor the health of the cluster and pods that run your application.



Deploying an AKS Cluster Using the Azure CLI

Azure Cloud Shell is a free, interactive shell that you can use to run the steps in this article. Common Azure tools are preinstalled and configured in Cloud Shell for you to use with your account. Just select the Copy button to copy the code, paste it in Cloud Shell, and then press Enter to run it. There are a few ways to open Cloud Shell:

- Select the Try It option in the upper-right corner of a code block.
- Open Cloud Shell in your browser.
- Select the Cloud Shell button on the menu in the upper-right corner of the Azure portal.



Deploying an AKS Cluster Using the Azure CLI

Create a **Resource Group** using the *az group create* command

The following example creates a resource group named *myResourceGroup* in the *eastus* location.

In Azure CLI, use the following command:

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

The following example output shows that the resource group has been created successfully.

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

Deploying an AKS Cluster Using the Azure CLI

Creating the AKS Cluster

Use the *az aks create* command to create an AKS cluster. Following example creates a cluster named *myAKSCluster* with one node. Azure Monitor for containers is also enabled using the --enable-addons monitoring parameter.

In Azure CLI, use the following command:

```
az aks create \
--resource-group myResourceGroup \
--name myAKSCluster \
--node-count 1 \
--enable-addons monitoring \
--generate-ssh-keys
```

After a few minutes, the command completes and returns JSON-formatted information about the cluster.

Deploying an AKS Cluster Using the Azure CLI

Connecting to the Cluster

To manage a Kubernetes cluster, you use kubectl, the Kubernetes command-line client. If you use Azure Cloud Shell, kubectl is already installed.

To install kubectl locally, in Azure CLI, use the following command:

```
az aks install-cli
```

```
az aks get-credentials --resource-group myResourceGroup --name myAKSCluster //*command downloads credentials and configures the Kubernetes CLI *
```

```
kubectl get nodes //* to return a list of the cluster nodes*
```

Deploying an AKS Cluster Using the Azure CLI

The following example output shows the single node that has been created in the previous steps. Make sure that the status of the node is **Ready**

```
NAME                        STATUS   ROLES   AGE    VERSION
aks-nodepool1-31718369-0    Ready    agent   6m44s  v1.9.11
```

Next step is to run the application.

Create a file named ***azure-vote.yaml*** and copy in the following YAML definition. If you use the Azure Cloud Shell, this file can be created by using ***vi*** or ***nano*** just like the way you create on virtual or physical systems. You can find the YAML Commands in the next slides.

Deploying an AKS Cluster Using the Azure CLI

```
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:
      containers:
      - name: azure-vote-
back
        image: redis
        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:
      containers:
      - name: azure-vote-
front
```

Deploying an AKS Cluster Using the Azure CLI

```
 image: microsoft/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
```

(Use the commands in the sequence)

Deploying an AKS Cluster Using the Azure CLI

Deploy the application using the *kubectl apply* command and specify the name of your YAML manifest

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

The following example output shows that the Deployments and Services have been created successfully.

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

Test the application: To monitor the progress, use the ***kubectl get service*** command with the ***--watch*** argument.

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

Deploying an AKS Cluster Using the Azure CLI

Initially, the EXTERNAL-IP for the azure-vote-front service is shown as pending.

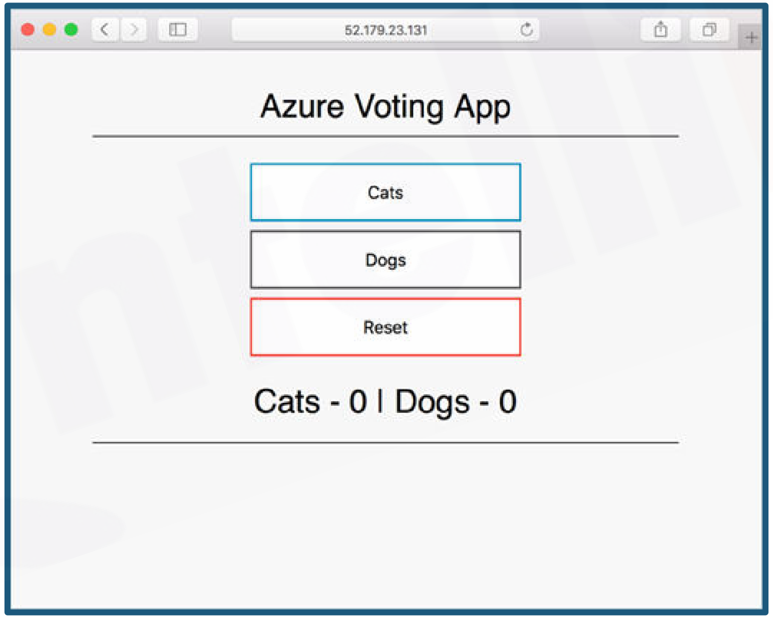
```
NAME               TYPE           CLUSTER-IP    EXTERNAL-IP   PORT(S)        AGE
azure-vote-front   LoadBalancer   10.0.37.27    <pending>     80:30572/TCP   6s
```

When the EXTERNAL-IP address changes from pending to an actual public IP address, use CTRL-C to stop the kubectl watch process. The following example output shows that a valid public IP address is assigned to the service.

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

Deploying an AKS Cluster Using the Azure CLI

To see the Azure Vote app in action, open a web browser to the external IP address of your service.



Deploying an AKS Cluster Using the Azure CLI

Monitoring Health and Logs

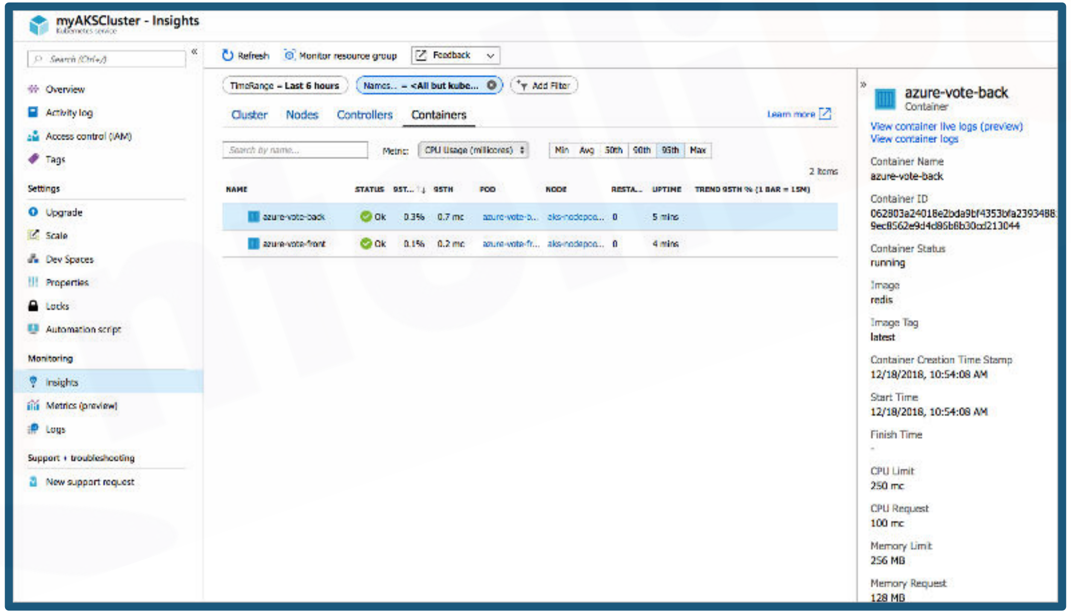
When the AKS cluster was created, Azure Monitor for containers was enabled to capture health metrics for both cluster nodes and pods. These health metrics are available in the Azure portal.

To see current status, uptime, and resource usage for the Azure Vote pods, complete the following steps:

- Open a web browser to the Azure portal https://portal.azure.com
- Select your resource group, such as *myResourceGroup*, and then select your AKS cluster, such as *myAKSCluster*
- Under Monitoring on the left-hand side, choose **Insights**
- Across the top, choose **+ Add Filter**
- Select Namespace as the property, then choose ***<All but kube-system>***
- Choose View the Containers

Deploying an AKS Cluster Using the Azure CLI

The *azure-vote-back* and *azure-vote-front* containers are displayed, as shown in the following example:



NAME	STATUS	95T...	95TH	POD	NODE	RESTA...	UPTIME	TREND 95TH % (1 BAR = 15M)
azure-vote-back	Ok	0.3%	0.7 mc	azure-vote-b...	aks-nodepoo...	0	5 mins	
azure-vote-front	Ok	0.1%	0.2 mc	azure-vote-fr...	aks-nodepoo...	0	4 mins	

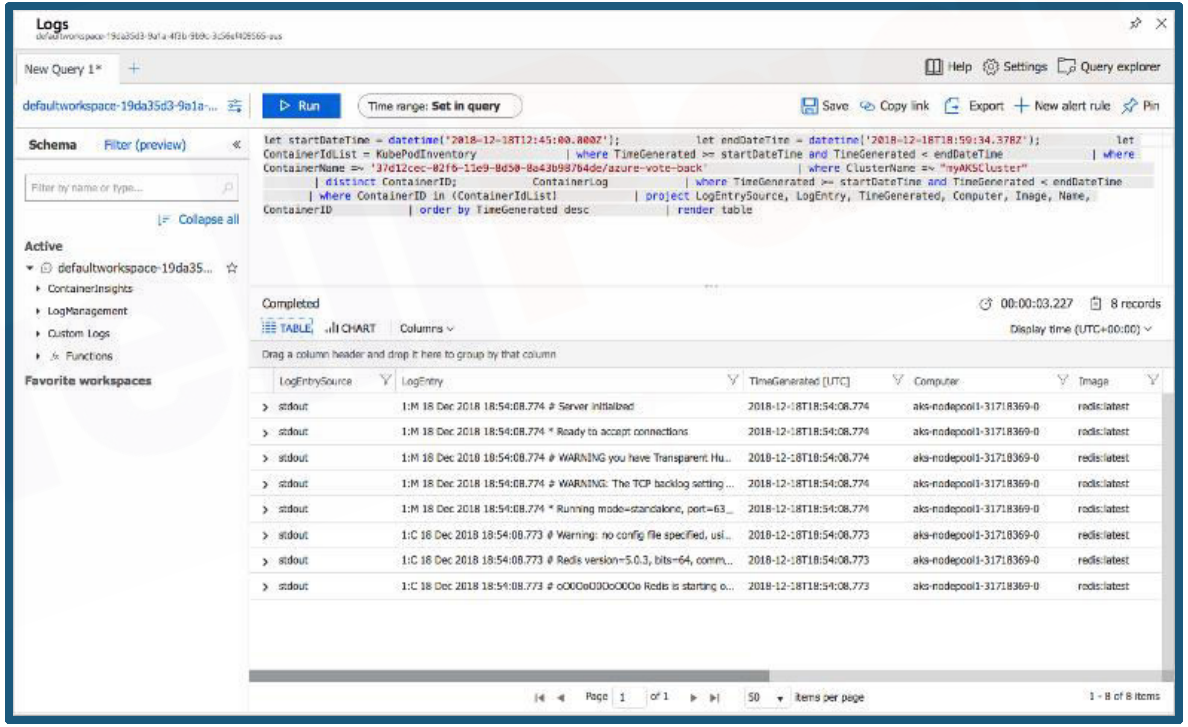
azure-vote-back
Container

View container live logs (preview)
View container logs

Container Name	azure-vote-back
Container ID	062803a24018e2bda9bf4353bfa2393488 9ec8562e9d4d85b8b30cd213044
Container Status	running
Image	redis
Image Tag	latest
Container Creation Time Stamp	12/18/2018, 10:54:08 AM
Start Time	12/18/2018, 10:54:08 AM
Finish Time	-
CPU Limit	250 mc
CPU Request	100 mc
Memory Limit	256 MB
Memory Request	128 MB

Deploying an AKS Cluster Using the Azure CLI

To see logs for the ***azure-vote-front*** pod, select the View container logs link on the right-hand side of the containers list. These logs include the ***stdout*** and ***stderr*** streams from the container.



```
let startDateTime = datetime('2018-12-18T12:45:00.000Z');          let endDateTime = datetime('2018-12-18T18:59:34.378Z');          let
ContainerIdList = KubePodInventory            | where TimeGenerated >= startDateTime and TimeGenerated < endDateTime           | where
ContainerName =~ '37d12cec-02f6-11e9-8d50-8a43b98764de/azure-vote-back'               | where ClusterName =~ "myAKSCluster"
        | distinct ContainerID;            ContainerLog            | where TimeGenerated >= startDateTime and TimeGenerated < endDateTime
       | where ContainerID in (ContainerIdList)          | project LogEntrySource, LogEntry, TimeGenerated, Computer, Image, Name,
ContainerID            | order by TimeGenerated desc            | render table
```

LogEntrySource	LogEntry	TimeGenerated [UTC]	Computer	Image
stdout	1:M 18 Dec 2018 18:54:08.774 # Server initialized	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 * Ready to accept connections	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING you have Transparent Hu...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 # WARNING: The TCP backlog setting ...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:M 18 Dec 2018 18:54:08.774 * Running mode=standalone, port=63...	2018-12-18T18:54:08.774	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # Warning: no config file specified, usi...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # Redis version=5.0.3, bits=64, comm...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest
stdout	1:C 18 Dec 2018 18:54:08.773 # oO0OoO0OoO0Oo Redis is starting o...	2018-12-18T18:54:08.773	aks-nodepool1-31718369-0	redis:latest

Deploying an AKS Cluster Using the Azure CLI

Deleting the Cluster

When the cluster is no longer needed, use the ***az group delete*** command to remove the resource group, container service, and all related resources.

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

Getting the Code for This Demo

The related application code, Dockerfile, and the Kubernetes manifest file are available on GitHub.

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

Hands-on

Hands-on

- ❑ **Deploy an AKS cluster using the Azure portal**
  - ❑ Create an AKS cluster
  - ❑ Connect to the cluster
  - ❑ Run the application
  - ❑ Test the application
  - ❑ Monitor health and logs
  - ❑ Delete the cluster

The related application code, Dockerfile, and the Kubernetes manifest file are available on GitHub.
https://github.com/Azure-Samples/azure-voting-app-redis

Hint: You may perform similar steps for the Azure portal which are used in CLI.

Hands-on

QUIZ

Quiz 1

Azure Apps Service?

A	It is an azure application which provides you major cloud services
B	It is a comprehensive platform for building cloud-based applications
C	It helps in the maintenance of your application
D	A type of service which provides security to the cloud-based applications

Answer 1

Azure Apps Service?

A It is an azure application which provides you major cloud services

[✓ B] It is a comprehensive platform for building cloud-based applications

C It helps in the maintenance of your application

D A type of service which provides security to the cloud-based applications

Quiz 2

Which of the programming languages you can use to build a web app in Azure?

A	Python
B	R
C	Go
D	C++

Answer 2

Which of the programming languages you can use to build a web app in Azure?

[✓ A Python]

B R

C Go

D C++

Quiz 3

Which is the right option of an Azure cloud storage service?

A	Dropbox
B	Github
C	Blob Storage
D	Azure Storage

Answer 3

Which is the right option of an Azure cloud storage service?

A	Dropbox
B	Github
C	Blob Storage
D	Azure Storage

Quiz 4

Which one of them is a manual method of deploying web apps?

A	FTP
B	Dropbox
C	Onedrop
D	Azure Storage

Answer 4

Which one of them is a manual method of deploying web apps?

- [✓ A FTP]
- B Dropbox
- C Onedrop
- D Azure Storage

Quiz 5

To host a web application, which is the right option?

A	Azure Vms
B	Web Apps
C	Azure Cloud Services
D	All of the above

Answer 5

To host a web application, which is the right option?

- A Azure Vms
- B Web Apps
- C Azure Cloud Services
- [✓ D] All of the above

Quiz 6

Which is the right way to manage Function app in Azure portal?

A Use App Service Editor >> App Settings >> Use Console >> Advanced Tools >> Deployment options >> CORS >> Authentication >> API Definition

B Advanced Tools >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Authentication >> API Definition

C API Definition >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Advanced Tools >> Authentication

D API Definition >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Advanced Tools >> Authentication >> Deployment options

Answer 6

Which is the right way to manage Function app in Azure portal?

[✓ **A**] Use App Service Editor >> App Settings >> Use Console >> Advanced Tools >> Deployment options >> CORS >> Authentication >> API Definition

B Advanced Tools >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Authentication >> API Definition

C API Definition >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Advanced Tools >> Authentication

D API Definition >> Use App Service Editor >> App Settings >> Use Console >> Deployment options >> CORS >> Advanced Tools >> Authentication >> Deployment options

India: +91-7847955955

US: 1-800-216-8930 (TOLL FREE)

sales@intellipaat.com

24/7 Chat with Our Course Advisor