



# AZ-204T00A Learning Path 01: Implement Azure App Service web apps



# Agenda

---

- Explore Azure App Service
- Configure web app settings
- Scale apps in Azure App Service
- Explore Azure App Service deployment slots

# Module 1: Explore Azure App Service



# Learning objectives

- Describe Azure App Service key components and value.
- Explain how Azure App Service manages authentication and authorization.
- Identify methods to control inbound and outbound traffic to your web app.
- Deploy an app to App Service using Azure CLI commands.

# Introduction

- Azure App Service is an HTTP-based service for hosting web applications, REST APIs, and mobile back ends.
- Applications run and scale in both Windows and Linux-based environments

# Examine Azure App Service

## **Built-in scale support**

- Save costs and meet demand through scaling
- Scale out/in manually or automated based on metrics
- Scale up/down by changing the app service plan

## **Continuous integration/ deployment support**

- Azure DevOps
- GitHub
- Bitbucket
- FTP
- Local Git repository
- And others

## **Deployment slots**

- Target deployments to test or production environments
- Customize what settings are swapped between slots

# Examine Azure App Service plans

## App Service plans

- Define a set of compute resources
- Can run one or more apps in the same plan
- Can be scaled up or down at any time to meet compute or feature needs

## Usage tiers

- Shared: shared compute resources targeting dev/test
- Basic: dedicated compute targeting dev/test
- Standard: run production workloads
- Premium: Enhanced performance and scale
- Isolated: high-performance, security and isolation

## How does my app run and scale?

- Shared: apps receive CPU minutes on a shared VM instance and can't scale out
- Other tiers: apps run on all the VM instances configured in the App Service plan

# Deploy to App Service

Every development team has unique requirements for their deployment pipeline. App Service supports both automated and manual deployment.

## Automated deployment

- Azure DevOps
- GitHub
- Bitbucket

## Manual deployment

- Git
- CLI
- Zipdeploy
- FTP/S

## Deployment slots

- Target deployments to test or production environments
- Customize what settings are swapped between slots



# Explore authentication and authorization in App Service

- Built-in authentication and authorization support
  - Implement with low to no code changes in your web app
- Identity providers available by default
  - Microsoft Identity Platform
  - Facebook
  - Google
  - Twitter
  - Apple
  - OpenID Connect

# Discover App Service networking features

## Multitenant App Service networking features

- Inbound features
  - App-assigned address
  - Access restrictions
  - Service endpoints
- Private endpoints
  - Outbound features
  - Hybrid Connections
  - Gateway-required VNet Integration
  - VNet Integration

## Single-tenant networking

Azure App Service Environment hosts Isolated SKU plans directly in your Azure virtual network.

- **External:** Exposes the hosted apps by using an IP address that is accessible on the internet.
- **Internal load balancer:** Exposes the hosted apps on an IP address inside your virtual network.

# Exercise: Create a static HTML web app by using Azure Cloud Shell

- In this exercise, you deploy a basic HTML+CSS site to Azure App Service by using the Azure CLI **az webapp up** command.
- You then update the code and redeploy it by using the same command.

## Objectives

- Download the sample app
- Create the web app
- Update and redeploy the app

# Summary and knowledge check

In this module, you learned how to:

- Describe Azure App Service key components and value.
- Explain how Azure App Service manages authentication and authorization.
- Identify methods to control inbound and outbound traffic to your web app.
- Deploy an app to App Service using Azure CLI commands.

1

Which App Service plan category provides the maximum scale-out capabilities?

2

What networking feature of App Service can be used to control outbound network traffic?

# Module 2: Configure web app settings



# Learning objectives

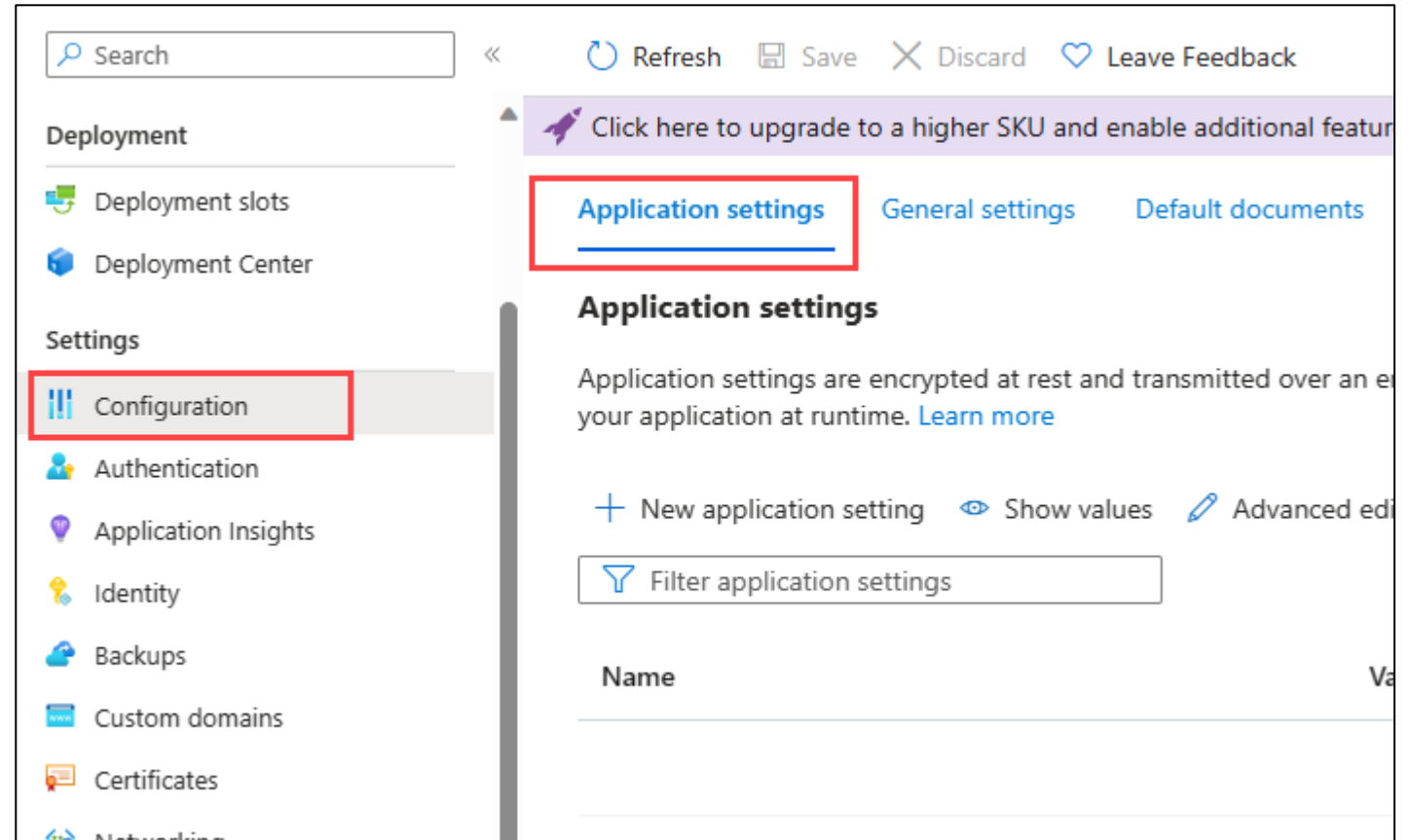
- Create application settings that are bound to deployment slots.
- Explain the options for installing TLS certificates for your app.
- Enable diagnostic logging for your app to aid in monitoring and debugging.
- Create virtual app to directory mappings.

# Introduction

- In App Service, app settings are passed as environment variables to the application code.
- For Linux apps and custom containers, App Service passes app settings to the container using the **--env** flag to set the environment variable in the container.

# Configure application settings ( 1 of 2 )

- Adding and editing settings
  - To add a new app setting, click **New application setting**.
  - To add or edit app settings in bulk, click the **Advanced** edit button.
- Configure connection strings
  - Adding and editing connection strings follow the same principles as other app settings and they can also be tied to deployment slots.





# Configure application settings ( 2 of 2 )

## Editing settings in bulk

```
[  
  {  
    "name": "name-1",  
    "value": "conn-string-1",  
    "type": "SQLServer",  
    "slotSetting": false  
  },  
  {  
    "name": "name-2",  
    "value": "conn-string-2",  
    "type": "PostgreSQL",  
    "slotSetting": false  
  },  
  ...  
]
```

# Configure general settings

## Stack settings

The software stack to run the app, including the language and SDK versions.

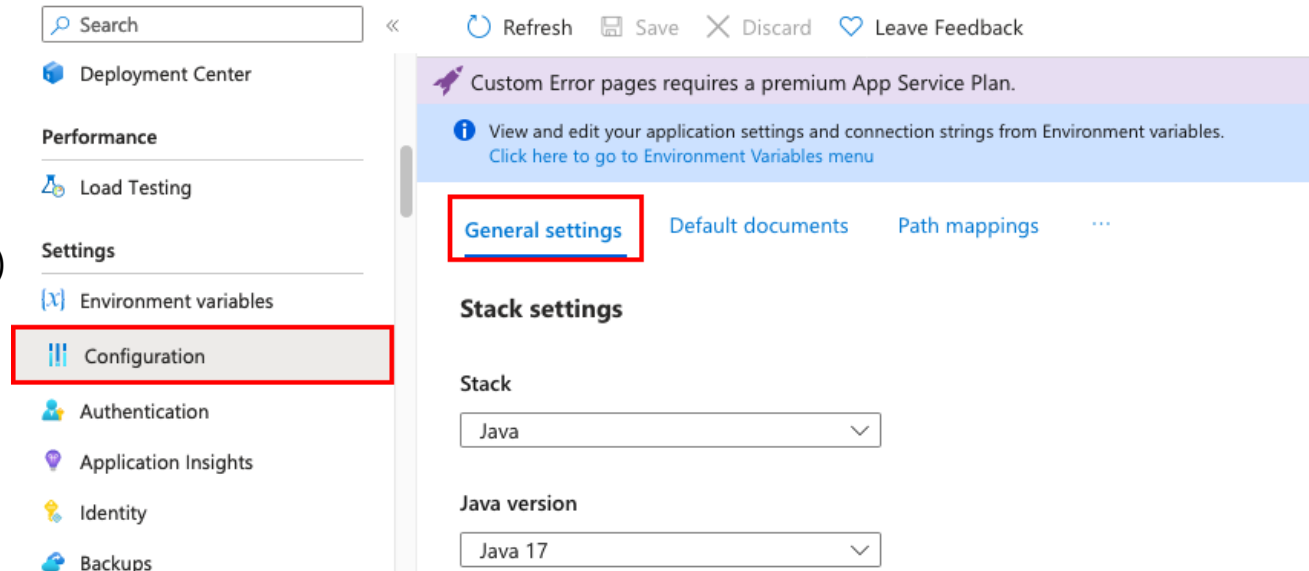
## Debugging

Enable remote debugging for ASP.NET , ASP.NET Core, or Node.js apps.

## Platform settings

Configure settings for the hosting platform, including:

- Bitness (32-bit or 64-bit)
- WebSocket protocol
- Always On
- Managed pipeline version
- HTTP version
- ARR affinity



# Configure path mappings

## Linux and containerized apps

- Add custom storage for your containerized app.
- Containerized apps include all Linux apps and Windows and Linux custom containers running on App Service.

## Windows apps (uncontainerized)

- Customize the IIS handler mappings and virtual applications and directories.
- Handler mappings enable adding custom script processors to handle requests for specific file extensions.

The screenshot shows the Microsoft Azure portal interface for configuring an App Service. The left sidebar contains a navigation menu with categories: Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Security, Events, Deployment, and Settings. The 'Configuration' option under Settings is highlighted with a red box. The main content area shows the 'Configuration' page for App Service, with tabs for Application settings, General settings, Default documents, and Path mappings (highlighted with a red box). The 'Path mappings' section includes a 'Handler mappings' subsection with a '+ New handler mapping' button and a table showing no handler mappings. Below this is the 'Virtual applications and directories' subsection with a '+ New virtual application or directory' button (highlighted with a red box) and a table listing existing mappings.

Virtual path	Physical Path	Type
/	site\wwwroot	Application
/api	site\wwwroot\api	Application

# Enable diagnostic logging

Type	Platform	Description
Application logging	Windows, Linux	Logs messages generated by your application code. The messages are generated by the web framework you choose.
Web server logging	Windows	Raw HTTP request data in the W3C extended log file format.
Detailed error logging	Windows	Copies of the <i>.html</i> error pages that would have been sent to the client browser.
Failed request tracing	Windows	Detailed tracing information on failed requests
Deployment logging	Windows, Linux	Deployment logging happens automatically and there are no configurable settings for deployment logging.

# Configure security certificates

- Options for adding certificates in App Service
  - Free App Service managed certificate
  - Purchase an App Service certificate
  - Import a certificate from Key Vault
  - Upload a private certificate
  - Upload a public certificate

# Summary and knowledge check

In this module, you learned how to:

- Create application settings that are bound to deployment slots.
- Explain the options for installing SSL/TLS certificates for your app.
- Enable diagnostic logging for your app to aid in monitoring and debugging.
- Create virtual app to directory mappings.

1

Which app configuration settings category is used to set the language and SDK version?

2

What types of diagnostic logging are supported on the Linux platform?

# Module 3: Scale apps in Azure App Service



# Learning objectives

- Identify scenarios for which autoscaling is an appropriate solution
- Create autoscaling rules for a web app
- Monitor the effects of autoscaling



# Introduction

- Autoscaling adjusts the number of available instances to meet the varying demands on your application
- Create rules to specify the conditions where instances should be added or removed
- Control costs

# Examine autoscale factors

- Autoscaling adjusts available resources based on the current demand.
- Autoscaling performs scaling in and out, as opposed to scaling up and down.
- Monitors the resource metrics of a web app as it runs and detects when additional resources are required based on the set conditions.
- When should you consider autoscaling?
  - Autoscaling provides elasticity for your services.
  - Autoscaling improves availability and fault tolerance.
  - Autoscaling isn't the best approach to handling long-term growth.

# Identify autoscale factors

## Autoscale conditions

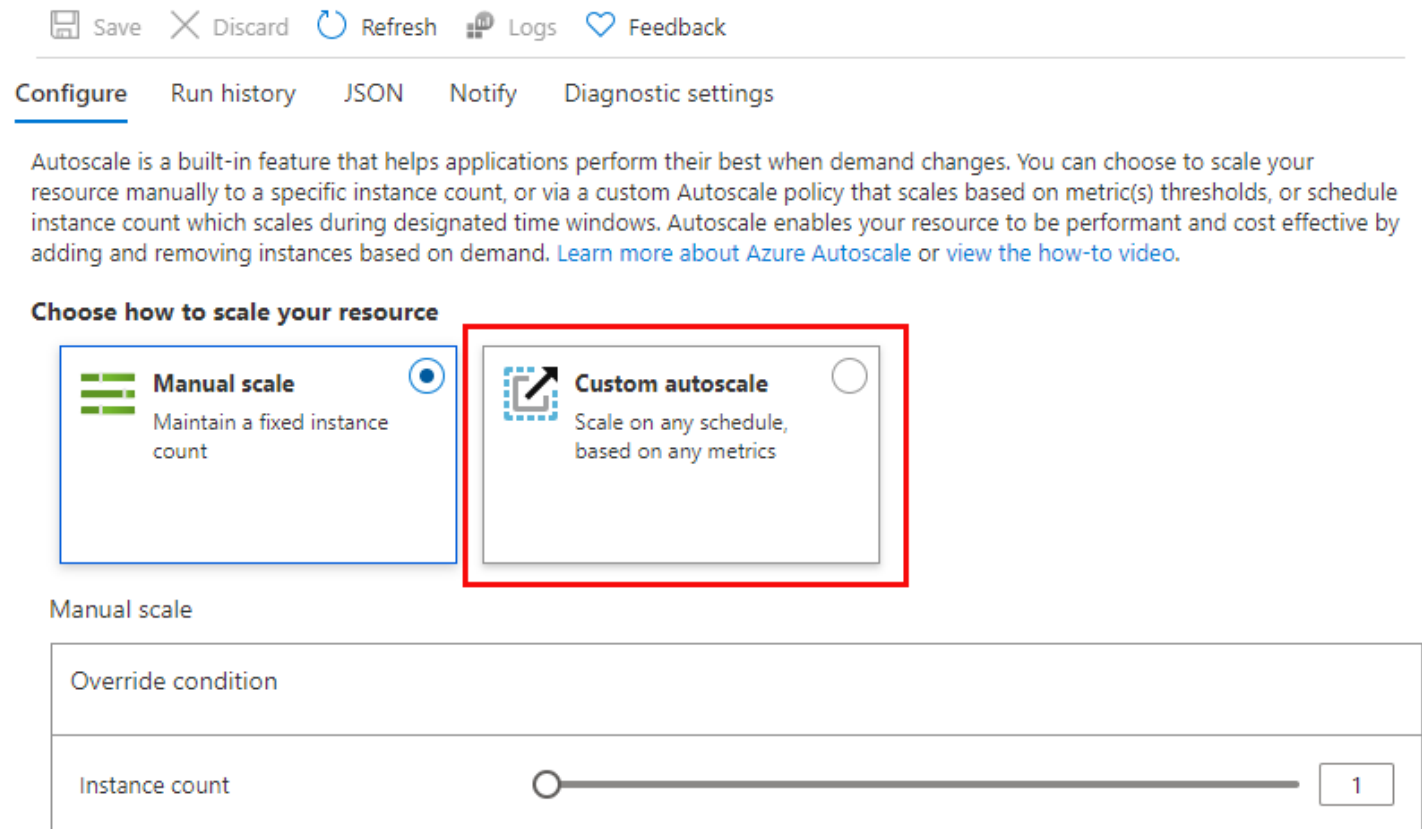
- Scale based on a metric, such as CPU usage or the length of a disk queue.
- Scale based on a schedule, such as a time of day or day of the week.
- Can create multiple conditions to handle complex needs.

## Autoscale metrics

- Metrics are based on all instances of an app
- CPU percentage
- Memory percentage
- Disk queue length – outstanding I/O requests
- HTTP queue length – number of client requests
- Data in – number of bytes received
- Data Out – number of bytes sent

# Enable autoscale in App Service

- Enable autoscaling
  - Not all pricing tiers support autoscaling
  - Some are only single instance or limited to manual scaling
- Add scale conditions
  - A default scale condition is created
  - Edit the default or create additional conditions
- Create scale rules
  - Conditions can contain one or more scale rules
  - Rules can be set based on calendar, metric, or both
- Monitor autoscaling behavior
  - View autoscale changes on the **Run history** chart
  - Tracks number of instances and which condition triggered the change



The screenshot shows the 'Configure' tab of the Azure App Service Autoscale settings. At the top, there are buttons for 'Save', 'Discard', 'Refresh', 'Logs', and 'Feedback'. Below these are tabs for 'Configure', 'Run history', 'JSON', 'Notify', and 'Diagnostic settings'. The main text explains that Autoscale is a built-in feature for scaling applications based on demand, metrics, or schedules. It then presents two options: 'Manual scale' (selected with a radio button) and 'Custom autoscale' (highlighted with a red box). The 'Manual scale' section is expanded, showing an 'Override condition' field and an 'Instance count' slider set to 1.

Save Discard Refresh Logs Feedback

Configure Run history JSON Notify Diagnostic settings

Autoscale is a built-in feature that helps applications perform their best when demand changes. You can choose to scale your resource manually to a specific instance count, or via a custom Autoscale policy that scales based on metric(s) thresholds, or schedule instance count which scales during designated time windows. Autoscale enables your resource to be performant and cost effective by adding and removing instances based on demand. [Learn more about Azure Autoscale](#) or [view the how-to video](#).

**Choose how to scale your resource**

**Manual scale** ☒ Maintain a fixed instance count

**Custom autoscale** ☐ Scale on any schedule, based on any metrics

Manual scale

Override condition

Instance count

# Enable autoscale in App Service

Scale rule

Metric source

Current resource

Resource type

App Service plans

Resource

fruitapi

Criteria

Metric namespace \*

Standard metrics

Metric name

CPU Percentage

1 minute time grain

Dimension Name

Instance

Operator

=

Dimension Values

All values

Add

If you select multiple values for a dimension, autoscale will aggregate the metric across the selected values, not evaluate the metric for each values individually.

CpuPercentage (Average)

--

☐ Enable metric divide by instance count

Operator \*

Greater than

Metric threshold to trigger scale action \*

70

%

Duration (minutes) \*

10

Time grain (minutes)

1

Add

Scale rule

Metric source

Current resource

Resource type

App Service plans

Resource

fruitapi

Criteria

Metric namespace \*

Standard metrics

Metric name

CPU Percentage

1 minute time grain

Dimension Name

Instance

Operator

=

Dimension Values

All values

Add

If you select multiple values for a dimension, autoscale will aggregate the metric across the selected values, not evaluate the metric for each values individually.

CpuPercentage (Average)

--

☐ Enable metric divide by instance count

Operator \*

Greater than

Metric threshold to trigger scale action \*

70

%

Duration (minutes) \*

10

Time grain (minutes)

1

Add

Scale based on a metric

Scale to a specific instance count

Scale is based on metric trigger rules but no rule(s) is defined; click [Add a rule](#) to create a rule. For example: 'Add a rule that increases instance count by 1 when CPU Percentage is above 70%'. If no rules is defined, the resource will be set to default instance count.

Minimum \*

1

Maximum \*

2

Default \*

1

Specify start/end dates

Repeat specific days

(UTC-08:00) Pacific Time (US & Canada)

06/07/2024

12:00:00 AM

06/07/2024

11:59:00 PM

Search (Ctrl+ /)

Overview

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Security

Deployment

Quickstart

Deployment slots

Deployment Center

Settings

Application settings

Configuration (Preview)

Authentication / Authorizati...

Application Insights

Identity

Backups

Custom domains

SSL settings

Networking

Scale up (App Service plan)

Scale out (App Service plan)

WebJobs

Push

MySQL In App

Save

Discard

Disable autoscale

Refresh

Configure

Run history

JSON

Notify

Observed instance count - this chart plots the instance count as observed by the auto scale engine. If the chart is empty it either means auto scale is in cool down period or auto scale was disabled over a period of time or auto scale was not configured.

Show data for last

1 hour

6 hours

12 hours

1 day

7 days

Custom

Pin to dashboard

Observed Capacity (Avg)

ScaleOnCPU

2.22

Autoscale events for this time range

View more details in the Activity Log

OPERATION	STATUS	EVENT CATEGORY	TIME	TIME STAMP	SUB...	EVENT INITIATED BY	RES...	RES...
1	Succeeded	Autoscale	20 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	20 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	26 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	26 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	32 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	32 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	38 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...
1	Succeeded	Autoscale	38 min ago	Sun Mar 17 2...	Azu...	Microsoft.Insights/autoscale...	Mic...	serv...

# Explore autoscale best practices

- Ensure the maximum and minimum values are different and have an adequate margin between them
- Choose the appropriate statistic for your diagnostics metric
- Choose the thresholds carefully for all metric types
- Check for conflicts when multiple rules are configured in a condition
- Always select a safe default instance count
- Configure autoscale notifications

# Summary and knowledge check

In this module, you learned how to:

- Identify scenarios for which autoscaling is an appropriate solution
- Create autoscaling rules for a web app
- Monitor the effects of autoscaling

1

How would you describe autoscaling?

2

Can you describe a scenario that is a suitable candidate for autoscaling?

# Module 4: Explore Azure App Service deployment slots





# Learning objectives

- Describe the benefits of using deployment slots
- Understand how slot swapping operates in App Service
- Perform manual swaps and enable auto swap
- Route traffic manually and automatically

# Introduction

- Deployment slots enable you to preview, manage, test, and deploy different development environments.
- Deployment slots are live apps with their own host names.
- App content and configuration elements can be swapped between two slots.

# Explore staging environments

- You can use a separate deployment slot instead of the default production slot when you're running in the **Standard**, **Premium**, or **Isolated** App Service plan tier.
- Deploying to a non-production slot has the following benefits:
  - You can validate app changes in a staging deployment slot before swapping it with the production slot.
  - Deploying an app to a slot first and swapping it into production makes sure that all instances of the slot are warmed up before being swapped into production.
  - After a swap, the slot with previously staged app now has the previous production app.

# Examine slot swapping

## When swapping slots, App Service does the following

- 1 Applies the following settings from the target slot to all instances of the source slot:
  - Slot-specific app settings and connection strings, if applicable.
  - Continuous deployment settings, if enabled.
  - App Service authentication settings, if enabled.
- 2 Wait for every instance in the source slot to complete its restart.
- 3 If local cache is enabled, trigger local cache initialization by making an HTTP request to the application root ("/") on each instance of the source slot.
- 4 If auto swap is enabled with custom warm-up, trigger Application Initiation by making an HTTP request to the application root ("/") on each instance of the source slot.
- 5 If all instances on the source slot are warmed up successfully, swap the two slots by switching the routing rules for the two slots.
- 6 Now that the source slot has the pre-swap app previously in the target slot, perform the same operation by applying all settings and restarting the instances.

# Swap deployment slots

- Swap deployment slots on your app's Deployment slots page and the Overview page.
- Configure auto swap
- Swap with preview
- Specify a custom warm-up
- Roll back and monitor a swap

# Route traffic in App Service

## Route production traffic automatically

- Go to your app's resource page and select **Deployment slots**.
- In the **Traffic %** column of the slot you want to route to, specify a percentage (between 0 and 100) to represent the amount of total traffic you want to route.

## Route production traffic manually

- In addition to automatic traffic routing, App Service can route requests to a specific slot.
- This is useful when you want your users to be able to opt in to or opt out of your beta app.
- To route production traffic manually, you use the **x-ms-routing-name** query parameter.

# Summary and knowledge check

In this module, you learned how to:

- Describe the benefits of using deployment slots
- Understand how slot swapping operates in App Service
- Perform manual swaps and enable auto swap
- Route traffic manually and automatically

1

What is the default routing rule applied to new deployment slots?

2

Do all configuration elements follow the content across a swap?

# Discussion and lab





# Group discussion questions

- What are deployment slots and how can they be used?
- How does autoscale work in App Service? How would you scale-up an application?
- Can you name the different options to deploy apps to App Service both manually and automatically?

# Lab 01: Build a web application on Azure platform as a service offerings

In this lab, you will explore how to create a web application on Azure by using the PaaS model. After the web application is created, you will learn how to upload existing web application files by using the Apache Kudu zip deployment option. You will then view and test the newly deployed web application.

<http://aka.ms/az204labs>

- Exercise 1: Build a backend API by using Azure Storage and the Web Apps feature of Azure App Service
- Exercise 2: Build a front-end web application by using Azure Web Apps

# End of presentation

