

Configure Azure App Service

1. Introduction

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/1-introduction>

Introduction

Completed

Azure Administrators are interested in solutions that make it easier to deploy and manage their web, mobile, and API applications. Azure Administrators are interested in solutions that are AI-ready.

Your company provides consumer research, and your team manages the on-premises servers. The servers you administer run the entire company infrastructure from web servers to databases. The hardware is aging and starting to struggle to keep up with some of the new data analysis applications. Rather than upgrade the hardware, the company decided to deploy Azure App Service.

In this module, you learn how to configure and manage Azure App Service. You learn about configuration settings, deployment slots, and custom domain names. You learn about application backup, recovery, and monitoring.

The goal of this module is to provide you with the knowledge and skills to effectively use Azure App Services.

Learning objectives

In this module, you learn how to:

- Identify features and usage cases for Azure App Service.
- Create an app with App Service.
- Configure deployment settings, specifically deployment slots.
- Secure your App Service app.
- Configure custom domain names.
- Back up and restore your App Service app.
- Configure Azure Application Insights.

Skills measured

The content in the module helps you prepare for [Exam AZ-104: Microsoft Azure Administrator](#).

Prerequisites

- Working knowledge of the Azure portal, so you can configure the service.
- Familiarity with cloud-based services, specifically web hosting services.

2. Implement Azure App Service

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/2-implement>

Implement Azure App Service

Completed

[Azure App Service](#) brings together everything you need to create websites, mobile backends, and web APIs for any platform or device. Applications run and scale with ease in both Windows and Linux-based environments.

App Service provides Quickstarts for programming languages. These languages include: ASP.NET, Java, Node.js, Python, and PHP.

App Service benefits

There are many advantages to using App Service to develop and deploy your web, mobile, and API apps. Review the following table and think about what features can help you host your App Service instances.

Benefit	Description
Multiple languages and frameworks	App Service has first-class support for ASP.NET, Java, Node.js, PHP, and Python. You can also run PowerShell and other scripts or executables as background services.
DevOps optimization	App Service supports continuous integration and deployment with Azure DevOps, GitHub, BitBucket, Docker Hub, and Azure Container Registry. You can promote updates through test and staging environments. Manage your apps in App Service by using Azure PowerShell or the cross-platform command-line interface (CLI).
Global scale with high availability	App Service helps you scale up or out manually or automatically. You can host your apps anywhere within the Microsoft global datacenter infrastructure, and the App Service SLA offers high availability.

Benefit	Description
Security and compliance	App Service is ISO, SOC, and PCI compliant. You can authenticate users with Microsoft Entra ID or with social logins via Google, Facebook, X, or Microsoft. Create IP address restrictions and manage service identities.
Application templates	Choose from an extensive list of application templates in Azure Marketplace, such as WordPress, Joomla, and Drupal.
Visual Studio integration	App Service offers dedicated tools in Visual Studio to help streamline the work of creating, deploying, and debugging.
API and mobile features	App Service provides turn-key CORS support for RESTful API scenarios. You can simplify your mobile app scenarios by enabling authentication, offline data sync, push notifications, and more.

3. Create an app with App Service

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/3-create-app-service>

Create an app with App Service

Completed

You can use the Web Apps, Mobile Apps, or API Apps features of Azure App Service, and create your own apps in the Azure portal.

Things to know about configuration settings

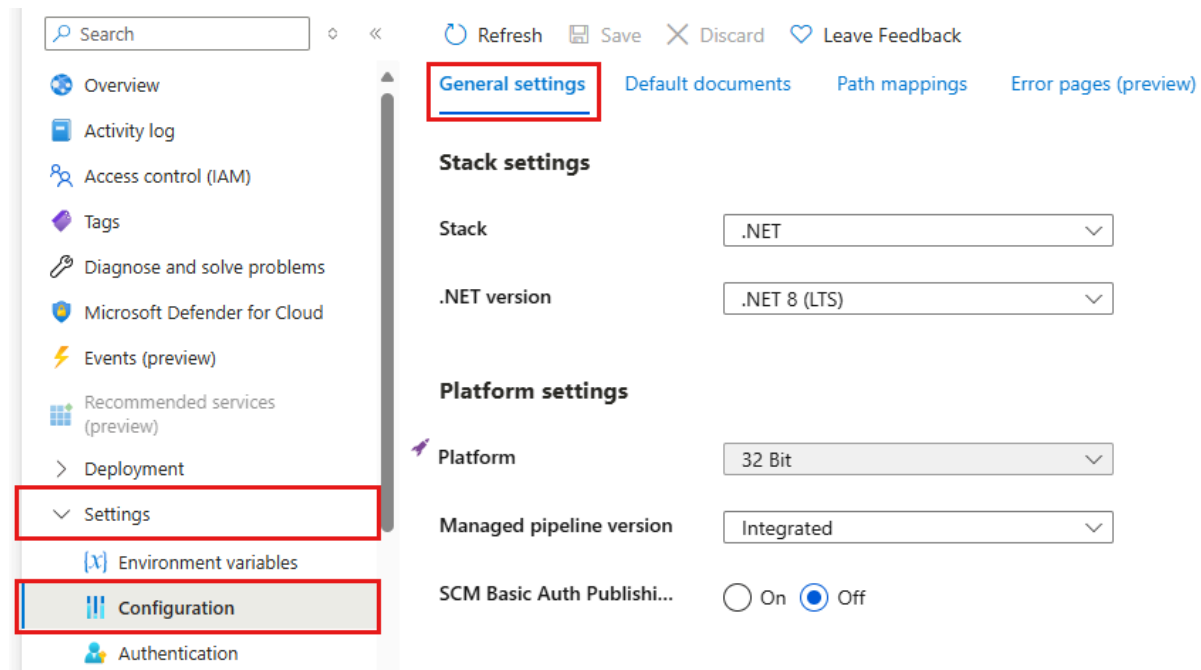
Let's examine some of the basic configuration settings you need to create an app with App Service.

- **Name:** The name for your app must be unique. The name identifies and locates your app in Azure. An example name is `webappces1.azurewebsites.net`. You can map a custom domain name, if you prefer to use that option instead.
- **Publish:** App Service hosts (publishes) your app as code or as a Docker Container.
- **Runtime stack:** App Service uses a software stack to run your app, including the language and SDK versions. For Linux apps and custom container apps, you can set an optional start-up command or file. Your choices for the stack include .NET Core, .NET Framework, Node.js, PHP, and Python. Various versions of each product are available for Linux and Windows.
- **Operating system:** The operating system for your app runtime stack can be Linux or Windows.

- **Region:** The region location that you choose for your app affects the App Service plans that are available.
- **Pricing plans:** Your app needs to be associated with an Azure App Service plan to establish available resources, features, and capacity. You can choose from pricing tiers that are available for the region location you selected.

Post-creation settings

After your app is created, other **Configuration** settings become available in the Azure portal, including app deployment options and path mapping.



Some of the extra configuration settings can be included in the developer's code, while others can be configured in your app. Here are a few of the extra application settings.

- **Always On:** You can keep your app loaded even when there's no traffic. This setting is required for continuous WebJobs or for WebJobs that are triggered by using a CRON expression.
- **Session affinity:** In a multi-instance deployment, you can ensure your app client is routed to the same instance for the life of the session.
- **HTTPS Only:** When enabled, all HTTP traffic is redirected to HTTPS.

Tip

Consider practicing on your own with the [Exercise - Create a web app in the Azure portal](#). This exercise provides a sandbox.

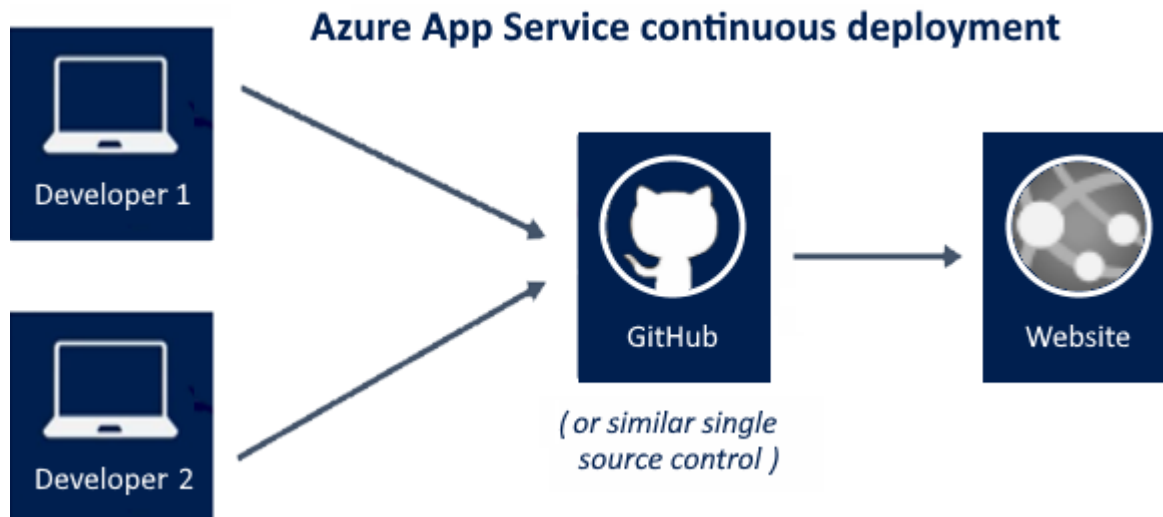
4. Explore continuous integration and deployment

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/4-explore-continuous-integration-on-deployment>

Explore continuous integration and deployment

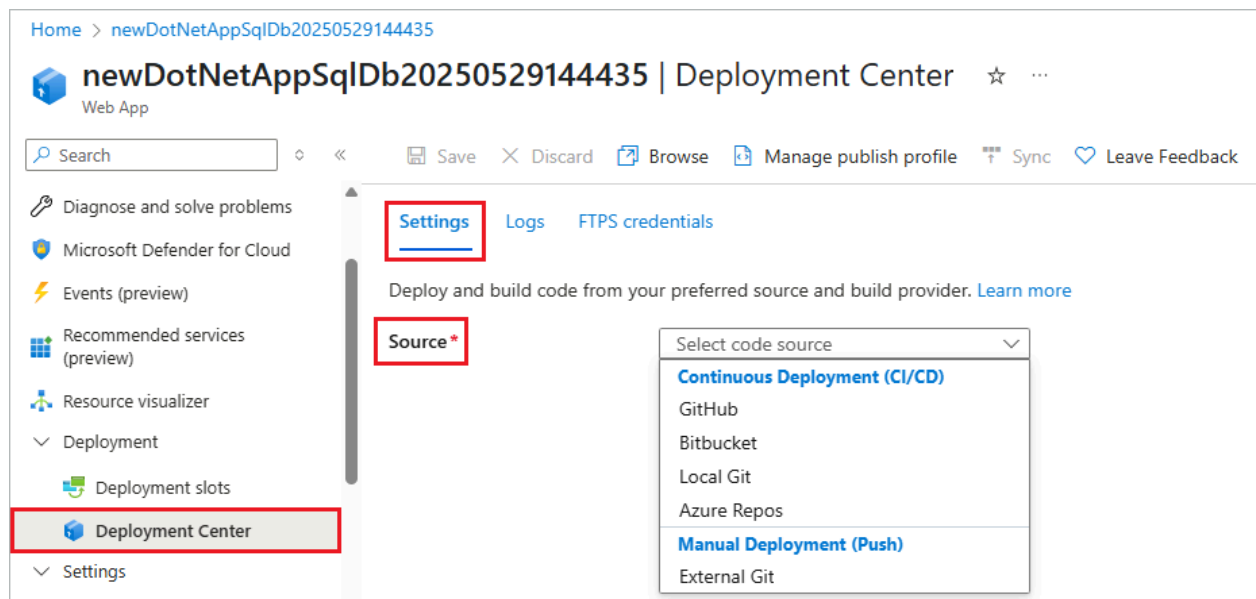
Completed

The Azure portal provides out-of-the-box continuous integration and deployment with Azure DevOps services, GitHub, Bitbucket, FTP, or a local Git repository on your development machine. You can connect your web app with any of the above sources and App Service handles the rest for you. App Service autosynchronizes your code and any future changes to the code into your web app. With Azure DevOps services, you can also define your own build and release process. Compile your source code, run tests, and build and deploy the release into your web app every time you commit the code. All of the operations happen implicitly without any need for human administration.



Things to know about continuous and manual deployment

When you create your web app with App Service, you can choose continuous or manual deployment. As you review these options, consider which deployment method to implement for your App Service apps. These options are located in the Deployment Center.



Continuous deployment (CI/CD) is a process used to push out new features and bug fixes in a fast and repetitive pattern with minimal impact on end users. Azure supports automated deployment directly from several sources:

- **GitHub:** Azure supports automated deployment directly from GitHub. Azure supports automated deployment directly from GitHub using two build providers. When you connect your GitHub repository to Azure, you can choose between **GitHub Actions** (default) and **App Service Build Service**.
- **Bitbucket:** With its similarities to GitHub, you can configure an automated deployment with Bitbucket.
- **Local Git:** The App Service Web Apps feature offers a local URL that you can add as a repository.
- **Azure Repos:** Azure Repos is a set of version control tools that you can use to manage your code. Whether your software project is large or small, using version control as soon as possible is a good idea.

Manual deployment enables you to manually push your code to Azure.

- **Remote Git:** The App Service Web Apps feature offers a Git URL that you can add as a remote repository. Pushing to the remote repository deploys your app.

5. Create deployment slots

Create deployment slots

Completed

When you deploy your web app, web app on Linux, mobile backend, or API app to Azure App Service, you can use a separate deployment slot instead of the default production slot.

Things to know about deployment slots

Let's take a closer look at the characteristics of deployment slots.

- Deployment slots are live apps that have their own hostnames.
- Deployment slots are available in the Standard, Premium, and Isolated v2 App Service pricing tiers. Your app needs to be running in one of these tiers to use deployment slots.
- The Standard, Premium, and Isolated tiers offer different numbers of deployment slots.
- App content and configuration elements can be swapped between two deployment slots, including the production slot.



Deployment Slots

Deployment slots are live apps with their own hostnames. App content and configurations elements can be swapped between two deployment slots, including the production slot.

NAME		STATUS	APP SERVICE PLAN	TRAFFIC %
webappces	PRODUCTION	Running	ASP-webapprg-a247	100
webappces-Staging		Running	ASP-webapprg-a247	0

Things to consider when using deployment slots

There are several advantages to using deployment slots with your App Service app. Review the following benefits and think about how they can support your App Service implementation.

- **Consider validation.** You can validate changes to your app in a staging deployment slot before swapping the app changes with the content in the production slot.
- **Consider reductions in downtime.** Deploying an app to a slot first and swapping it into production ensures that all instances are ready. This option eliminates downtime when you deploy your app. The traffic redirection is seamless, and no requests are dropped because of swap operations. The entire workflow can be automated by configuring **Auto swap** when preswap validation isn't needed.
- **Consider restoring to last known good site.** After a swap, the slot with the previously staged app now has the previous production app. If the changes swapped into the production slot aren't as you expected, you can perform the same swap immediately to return to your "last known good site."
- **Consider Auto swap.** Auto swap streamlines Azure Pipeline scenarios where you want to deploy your app continuously with zero cold starts and zero downtime for app customers. When Auto swap is enabled from a slot into production, every time you push your code changes to that slot, App Service automatically swaps the app into production after it's warmed up in the source slot.

6. Add deployment slots

Add deployment slots

Completed

Deployment slots are configured in the Azure portal. You can swap your app content and configuration elements between deployment slots, including the production slot.

Things to know about creating deployment slots

Let's review some details about how deployment slots are configured.

- New deployment slots can be empty or cloned.
- Deployment slot settings fall into three categories:
 - Slot-specific app settings and connection strings (if applicable).
 - Continuous deployment settings (when enabled).
 - Azure App Service authentication settings (when enabled).
- When you clone a configuration from another deployment slot, the cloned configuration is editable. Some configuration elements follow the content across the swap. Other slot-specific configuration elements stay in the source slot after the swap.

Swapped settings versus slot-specific settings

The following table lists settings that are swapped between deployment slots. The table also lists settings that remain in the source slot (slot-specific). As you review these settings, consider which features are required for your App Service apps. Read more about [which settings are swapped](#).

Swapped settings	Slot-specific settings
Language stack and version, 32/64-bit App settings * Connection strings * Mounted storage accounts*	Custom domain names Nonpublic certificates and TLS/SSL settings Scale settings Always On

Swapped settings	Slot-specific settings
Public certificates	IP restrictions
WebJobs content	WebJobs schedulers
Hybrid connections **	Diagnostic settings
Service endpoints **	Cross-origin resource sharing (CORS)
Azure Content Delivery Network **	Virtual network integration
Path mapping	Managed identities

* Setting can be configured to be slot-specific.

** Feature isn't currently available.

7. Secure your App Service app

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/7-secure-app-service>

Secure your App Service app

Completed

Azure App Service provides built-in [authentication and authorization](#) support. You can sign in users and access data by writing minimal or no code in your web app, API, and mobile backend, and also your Azure Functions apps.

Secure authentication and authorization require deep understanding of security, including federation, encryption, JSON web tokens (JWT) management, grant types, and so on. App Service provides these utilities so you can spend more time and energy on providing business value to your customer.

Note

You aren't required to use Azure App Service for authentication and authorization. Many web frameworks are bundled with security features, and you can use your preferred service.

Things to know about app security with App Service

Let's take a closer look at how App Service helps you provide security for your app.

- The authentication and authorization security module in Azure App Service runs in the same environment as your application code, yet separately.
- The security module is configured by using app settings. No SDKs, specific languages, or changes to your application code are required.
- The security module handles several tasks for your app:
 - Authenticate users with the specified provider
 - Validate, store, and refresh tokens
 - Manage the authenticated session
 - Inject identity information into request headers

Things to consider when using App Service for app security

You configure authentication and authorization security in App Service by selecting features in the Azure portal. Review the following options and think about what security can benefit your App Service apps implementation.

- **Allow Anonymous requests (no action).** Defer authorization of unauthenticated traffic to your application code. For authenticated requests, App Service also passes along authentication information in the HTTP headers. This feature provides more flexibility for handling anonymous requests. With this feature, you can present multiple sign-in providers to your users.
- **Allow only authenticated requests.** Redirect all anonymous requests to `/.auth/login/<provider>` for the provider you choose. The feature is equivalent to **Log in with <provider>**. If the anonymous request comes from a native mobile app, the returned response is an `HTTP 401 Unauthorized` message. With this feature, you don't need to write any authentication code in your app.

Important

This feature restricts access to **all** calls to your app. Restricting access to all calls might not be desirable if your app requires a public home page, as is the case for many single-page apps.

- **Logging and tracing.** View authentication and authorization traces directly in your log files. If you see an authentication error that you didn't expect, you can conveniently find all the details by looking in your existing application logs. If you enable failed request tracing, you can see exactly how the security module participated in a failed request. In the trace logs, look for references to a module named `EasyAuthModule_32/64`.

8. Create custom domain names

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/8-create-custom-domain-names>

Create custom domain names

Completed

When you create a web app, Azure assigns the app to a subdomain of `azurewebsites.net`. Suppose your web app is named `contoso`. Azure creates a URL for your web app as `contoso.azurewebsites.net`. Azure also assigns a virtual IP address for your app. For a production web app, you might want users to see a custom domain name.

What is a custom domain?

A domain name is the address people type into a web browser to reach your website. A custom domain is a domain name that you own and configure to point to your Azure-hosted app, replacing the default Azure domain.

For example:

- Default Azure domain: `myapp-000000.westus.azurewebsites.net`
- Custom domain: `www.contoso.com`

Using a custom domain allows you to:

- Establish a branded, user-friendly web address.
- Improve trust and credibility with customers.
- Manage and secure traffic to your application.

Steps to configure a custom domain name for your app

Creating a custom domain name requires providers, security, and naming information.

Add custom domain



Domain provider * ⓘ

☐ App Service Domain

☒ All other domain services

TLS/SSL certificate * ⓘ

☒ App Service Managed Certificate

☐ Add certificate later

TLS/SSL type * ⓘ

☒ SNI SSL

☐ IP based SSL

Enter your custom domain. We'll give you hostname records to register with your domain provider, and we can validate and add your custom domain here. [Learn more](#)

Domain * ⓘ

www.contoso.com

Hostname record type * ⓘ

CNAME (www.example.com or any subd...

Domain validation

To validate your domain ownership, copy the hostname records below and enter them with your domain provider. [Learn more](#)

Export to CSV

Type	Host	Value	Status
CNAME	www	webapp51451367.azurewebsites.net	
TXT	asuid.www	0DE62CB0A68623CB52D87E9A38105C8F8CDE5432741E4CD63DBA9E141B495DAE	

There are three steps to create a custom domain name.

1. **Reserve your domain name.** The easiest way to set up a custom domain is to buy one directly in the Azure portal. (This name isn't the Azure assigned name of `*.azurewebsites.net`.) The registration process enables you to manage your web app's domain name directly in the Azure portal instead of going to a third-party site. Configuring the domain name in your web app is also a simple process in the Azure portal.
2. **Create DNS records to map the domain to your Azure web app.** The Domain Name System (DNS) uses data records to map domain names to IP addresses. There are several types of DNS records.

- For web apps, you create either an **A** (Address) record or a **CNAME** (Canonical Name) record.
 - An **A** record maps a domain name to an IP address.
 - A **CNAME** record maps a domain name to another domain name. DNS uses the second name to look up the address. Users still see the first domain name in their browser. As an example, you could map **contoso.com** to your **webapp.azurewebsites.net** URL.
- If the IP address changes, a **CNAME** entry is still valid, whereas an **A** record must be updated.
- Some domain registrars don't allow **CNAME** records for the root domain or for wildcard domains. In such cases, you must use an **A** record.

3. **Enable the custom domain.** After you have your domain and create your DNS record, use the Azure portal to validate your custom domain and add it to your web app. Be sure to test your domain before publishing.

Important

App Service offers free managed TLS certificates. Certificates auto-renew 30 days before expiry. In the Azure portal, go to **Custom domains** → **Add binding** → **App Service Managed Certificate**.

9. Back up and restore your App Service app

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/9-backup-app-service>

Back up and restore your App Service app

Completed

The **Backup and Restore feature** in Azure App Service lets you easily create backups manually or on a schedule. You can configure the backups to be retained for a specific or indefinite amount of time. You can restore your app or site to a snapshot of a previous state by overwriting the existing content or restoring to another app or site.

The **Backups** page lists all the automatic and custom backups for your app and displays the status of each.

my-demo-app | Backups

App Service

Search (Cmd+/)

Events (preview)

Deployment

Quickstart

Deployment slots

Deployment Center

Settings

Configuration

Authentication

Application Insights (preview)

Identity

Backups

Custom domains

TLS/SSL settings

TLS/SSL settings (preview)

Monitoring

Oldest backup

09/04/2022

Automatic backup

Every 1 hour

Type : All

Status : All

Time range : None

Add filter

Reset

Showing 10 of 206 results

Backup time ↓	Status ↑	Type ↑	Restore
09/05/2022, 14:52:19	✓ Succeeded	Automatic	↶
09/05/2022, 14:47:15	✓ Succeeded	Custom	↶
09/05/2022, 14:36:31	✗ Failed	Custom	↶
09/05/2022, 14:20:20	✗ Failed	Custom	↶
09/05/2022, 13:52:19	✓ Succeeded	Automatic	↶
09/05/2022, 12:52:19	✓ Succeeded	Automatic	↶
09/05/2022, 11:52:19	✓ Succeeded	Automatic	↶

Things to know about Backup and Restore

Examine the following details about the Backup and Restore feature. Think about how you can implement this feature for your App Service apps.

- Back up and restore is supported in the Basic, Standard, Premium, and Isolated tiers. For the Basic tier, you can only back up and restore the production slot.
- You need an Azure storage account and container in the same subscription as the app to back up.
- Azure App Service can back up the following information to the Azure storage account and container you configured for your app:
 - App configuration settings
 - File content
 - Any database connected to your app (SQL Database, Azure Database for MySQL, Azure Database for PostgreSQL, MySQL in-app)
- In your storage account, each backup consists of a Zip file and XML file:
 - The Zip file contains the back-up data for your app or site.
 - The XML file contains a manifest of the Zip file contents.
- You can configure backups manually or on a schedule.
- Full backups are the default.
- Partial backups are supported. You can specify files and folders to exclude from a backup.

- You restore partial backups of your app or site the same way you restore a regular backup.
- Backups can hold up to 10 GB of app and database content.
- Backups for your app or site are visible on the **Containers** page of your storage account and app (or site) in the Azure portal.

Things to consider when creating backups and restoring backups

Let's review some considerations about creating a backup for your app or site, and restoring data and content from a backup.

- **Consider full backups.** Do a full backup to easily save all configuration settings, all file content, and all database content connected with your app or site.

When you restore a full backup, all content on the site is replaced with whatever is in the backup. If a file is on the site, but not in the backup, the file is deleted.

- **Consider partial backups.** Specify a partial backup so you can choose exactly which files to back up.

When you restore a partial backup, any content located in an excluded folder or file is left as-is.

- **Consider browsing back-up files.** Unzip and browse the Zip and XML files associated with your backup to access your backups. This option lets you view the content without actually performing an app or site restore.
- **Consider firewall on back-up destination.** If your storage account is enabled with a firewall, you can't use the storage account as the destination for your backups.

10. Use Azure Application Insights

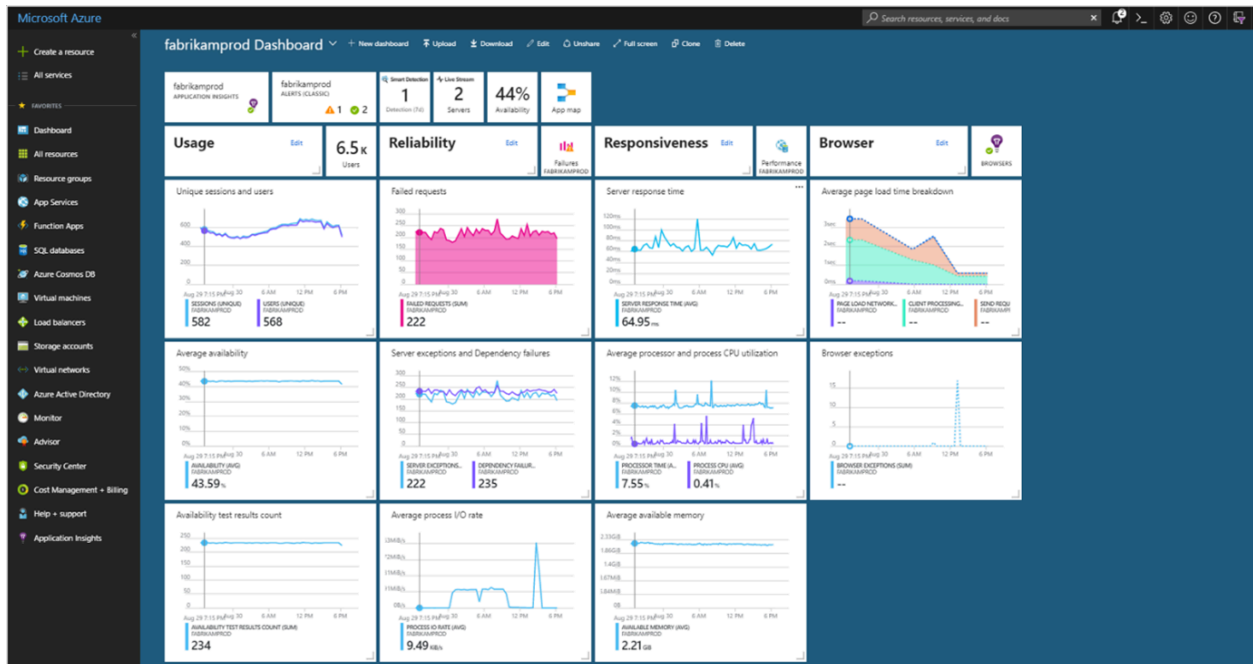
<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/10-use-application-insights>

Use Azure Application Insights

Completed

[Azure Application Insights](#) is a feature of Azure Monitor that lets you monitor your live applications. You can integrate Application Insights with your App Service configuration to automatically detect performance anomalies in your apps.

Application Insights is designed to help you continuously improve the performance and usability of your apps. The feature offers powerful analytics tools to help you diagnose issues and understand what users actually do with your apps.



Things to know about Application Insights

Let's examine some characteristics of Application Insights for Azure Monitor.

- Application Insights works on various platforms including .NET, Node.js, and Java EE.
- The feature can be used for configurations that are hosted on-premises, in a hybrid environment, or in any public cloud.
- Application Insights integrates with your Azure Pipeline processes, and has connection points to many development tools.

Things to consider when using Application Insights

Application Insights is ideal for supporting your development team. The feature helps developers understand how your app is performing and how it's being used. Consider monitoring the following items in your App Service configuration scenario.

- **Consider Request rates, response times, and failure rates.** Find out which pages are most popular, at what times of day, and where your users are. See which pages perform best. If your response times and failure rates go high when there are more requests, then perhaps you have a resourcing problem.
- **Consider Dependency rates, response times, and failure rates.** Use Application Insights to discover if external services are degrading your app performance.

- **Consider Exceptions.** Analyze the aggregated statistics, or pick specific instances and drill into the stack trace and related requests. Both server and browser exceptions are reported.
- **Consider Page views and load performance.** Collect the number of page views reported by your users' browsers and analyze the load performance.
- **Consider User and session counts.** Application Insights can help you keep track of the number of users and sessions connected to your app.
- **Consider Performance counters.** Add Application Insights performance counters from your Windows or Linux server machines. Monitor performance output for the CPU, memory, network usage, and so on.
- **Consider Host diagnostics.** Integrate diagnostics from Docker or Azure into your app Application Insights.
- **Consider Diagnostic trace logs.** Implement trace logs from your app to help correlate trace events with requests and diagnose issues.
- **Consider Custom events and metrics.** Write your own custom events and metric tracking algorithms as client or server code. Track business events such as number of items sold, or number of games won.

Tip

Consider extending your learning with the [Troubleshoot solutions by using Application Insights](#) training module.

11. Exercise: Implement Web Apps

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/11-simulation-web-apps>

Exercise: Implement Web Apps

Completed

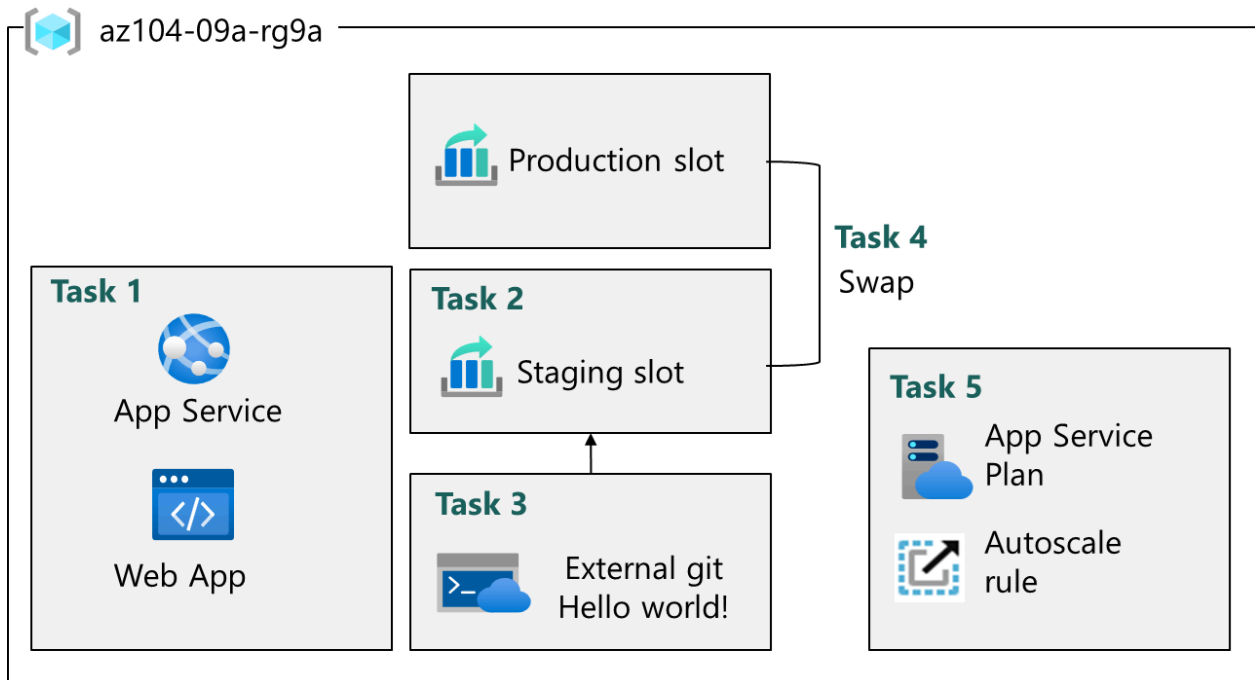
Lab scenario

Your organization is migrating on-premises web apps to Azure. As the Azure Administrator you need to:

- Host web sites running on Windows servers by using the PHP runtime stack.

- Use Azure Web Apps deployment slots.

Architecture diagram



Job skills

- Create an Azure web app.
- Create a staging deployment slot.
- Configure Web App deployment settings.
- Deploy code to the staging deployment slot.
- Swap the staging slots.
- Configure and test autoscaling of your Azure web app.

Note

Estimated time: 20 minutes. To complete this exercise, you need an [Azure subscription](#).

Launch the exercise, and follow the instructions. When finished, be sure to return to this page so you can continue learning.

[Launch Exercise](#)

12. Module assessment

Module assessment

Completed

13. Summary and resources

<https://learn.microsoft.com/en-us/training/modules/configure-azure-app-services/13-summary-resources>

Summary and resources

Completed

Azure App Service is an HTTP-based service for hosting web applications. With App Service, you can develop web apps in your favorite language. The service lets you easily run and scale your web apps on Windows and Linux-based environments.

In this module, you reviewed the features and usage cases for Azure App Service. You learned how to create, secure, and back up your web apps. You explored how to configure deployment settings, including deployment slots, and custom domain names for your web apps. You discovered how to use Azure Application Insights to monitor web apps.

The main takeaways for this module

- Azure App Service lets you develop and deploy web, mobile, and API apps.
- Azure App Service configuration settings include runtime stack, operating system, region, and App Service plan.
- Deployment slots help you manage different app stages. For example, development, test, stage, and production.
- The default Azure App Service domain name can be customized for your organization.
- Azure Application Insights is a feature of Azure Monitor that lets you monitor your live applications. You can integrate Application Insights with your App Service configure to automatically detect performance anomalies in your apps.
- Application Insights lets you continuously monitor the performance and usability of your apps.

Learn more with Copilot

Copilot can assist you in configuring Azure infrastructure solutions. Copilot can compare, recommend, explain, and research products and services where you need more information. Open a Microsoft Edge browser and choose Copilot (top right) or navigate to copilot.microsoft.com. Take a few minutes to try these prompts and extend your learning with Copilot.

- What are the main tasks to configure an Azure App Service web app?
- What options are available for scaling an Azure App Service web app?

Learn more with documentation

- [App Service overview](#). This article provides an overview of the App Service and why you would use this service.
- [Configure an App Service app](#). This article explains how to configure common settings for web apps, mobile back end, or API app.
- [Set up staging environments in Azure App Service](#). The article covers deployment slots and swap operations.

Learn more with self-paced training

- [Stage a web app deployment for testing and rollback by using App Service deployment slots](#). Learn to use deployment slots to streamline deployment and roll back.
- [Explore Azure App Service deployment slots](#). Learn how slot swapping works and how to route traffic to different slots.
- [Host a web application with Azure App Service](#). Learn how to create a website through the hosted web app platform in Azure App Service.