**Azure App Services - Web Apps**

* Introduction
* App Service - Application Types
* Deploy Web Apps
* App Service plans
* Configuring Web Apps
* Understanding Deployment Slots and Rollback deployments
* Horizontal and Vertical Scaling of a Web App.
* Configure auto-scale.
* Change the size of an instance.

**Azure App Services Introduction**

* Azure App Service is the cloud PaaS service that integrates everything you need to **quickly and easily** build web and mobile apps for **any platform** and **any device**.
* Built for developers, App Service is a **fully managed platform** with powerful capabilities such as built-in DevOps, continuous integration with Visual Studio Team Services and GitHub, staging and production support, and automatic patching.
* App Service is a **platform as a service (PaaS)**, which means that the OS and application stack are managed for you by Azure; you only manage your application and its data.
* Azure manages OS patching on two levels, the physical servers and the guest virtual machines (VMs) that run the App Service resources. Both are updated monthly. These updates are applied automatically, in a way that guarantees the high availability SLA of Azure services.
* New stable versions of supported language runtimes (major, minor, or patch) are periodically added to App Service instances. Some updates overwrite the existing installation, while others are installed side by side with existing versions. An overwrite installation means that your app automatically runs on the updated runtime. A side-by-side installation means you must manually migrate your app to take advantage of a new runtime version.

**App Types**

App Service allows you to create the following app types from a single development experience:

1. [**Web Apps**](https://azure.microsoft.com/en-in/documentation/articles/app-service-web-overview) - Quickly create and deploy mission critical Web apps that scale with your business.
2. [**API Apps**](https://azure.microsoft.com/en-in/documentation/articles/app-service-api-apps-why-best-platform) - Easily build and consume Cloud APIs.
3. [**Logic Apps**](https://azure.microsoft.com/en-in/documentation/articles/app-service-logic-what-are-logic-apps) - Automate the access and use of data across clouds without writing code.
4. **Function Apps –** Function written by developer and executed without any dedicated hardware.

As a single integrated service, App Service makes it easy to compose the above app types into a single solution, allowing you to easily build apps that target both **web and mobile clients** using the same back-end and integrate with on premise systems as well as popular services such as Office 365 and salesforce.com.

**App Service Web Apps** is a fully managed **compute platform** that is optimized for hosting websites and web applications. This [platform-as-a-service](https://en.wikipedia.org/wiki/Platform_as_a_service) (PaaS) offering of Microsoft Azure lets you focus on your business logic while Azure takes care of the infrastructure to run and scale your apps.

The compute resources may be on shared or dedicated virtual machines (VMs), depending on the pricing tier that you choose.

Your code can be in any language or framework that is supported by [Azure App Service](https://docs.microsoft.com/en-us/azure/app-service/app-service-value-prop-what-is), such as **ASP.NET, ASP.NET Core, .NET7, Node.js, Ruby, Java, PHP, or Python**.

**Why Web Apps:**

1. Multiple languages and frameworks
2. Application templates in Azure Marketplace. Templates in the Azure Marketplace, such as WordPress, Joomla, and Drupal.
3. DevOps optimization. Continuous integration and Continuous deployment.
4. Test in production using Deployment slots.
5. Global scale with high availability. Scale up or out manually or automatically.
6. Connections to SaaS platforms and on-premises data.
7. Visual Studio Integration.
8. App Service is ISO, SOC, and PCI compliant.

**Create and Deploy App Service Web Apps**

Deploying your app to App Service is a matter of deploying your code, binaries, content files, and their respective directory structure, to the **/site/wwwroot** directory in Azure.

1. **Create a Web App in Azure Portal**
   1. Login to Azure Portal, <https://portal.azure.com/>
   2. Azure Portal 🡪 Search App Services
   3. + Create 🡪 Select Web Apps
   4. Basic Tab:
      1. Resource Group="DemoRG",
      2. Name = "sandeep-webapp",
      3. Publish = Code,
      4. Operating System = Windows
      5. Region = East US
      6. Runtime stack = .NET 7 (STS)
      7. Windows Plan = Create new 🡪 Name=MyDemoRGEastUSPlan, Pricing plans 🡪 Standard S1

A screenshot of a computer

Description automatically generated

* 1. Deployment Tab
     1. Leave default 🡪 Next: Networking
  2. Networking Tab
     1. Leave default 🡪 Next: Monitoring >
  3. Monitoring Tab
     1. Enable Application Insights = Yes
     2. Application Insights = sandeep-webapp-appInsight
  4. Click Review + create -> Create

1. **Publishing the Sample Application:**
2. Clone the Application to local machine

C:\>mkdir demos

C:\>cd demos

C:\demos>git clone <https://github.com/sandeepsonihyd/HelloWorldApp.git>

C:\demos>cd HelloWorldApp

C:\demos\HelloWorldApp>dir

You have pulled a .net application from Git Repository

1. **Publish to Azure App Service**
2. **Azure Portal 🡪 App Services 🡪 sandeep-webapp** 🡪 **Deployment Center**
   * Source = Local Git
   * Save
   * Copy the Git Clone URL

A screenshot of a computer

Description automatically generated

* + Switch to Local Git/FTP credentials 🡪 Copy Local Git username and Password

A screenshot of a computer

Description automatically generated

**Switch to your local machine:**

1. C:\demos\HelloWorldApp>git remote remove origin
2. C:\demos\HelloWorldApp>git remote add origin <https://sandeep-webapp.scm.azurewebsites.net:443/sandeep-webapp.git>
3. C:\demos\HelloWorldApp>**git push origin master**
4. Use the credentails copied in step (a)

A screenshot of a computer

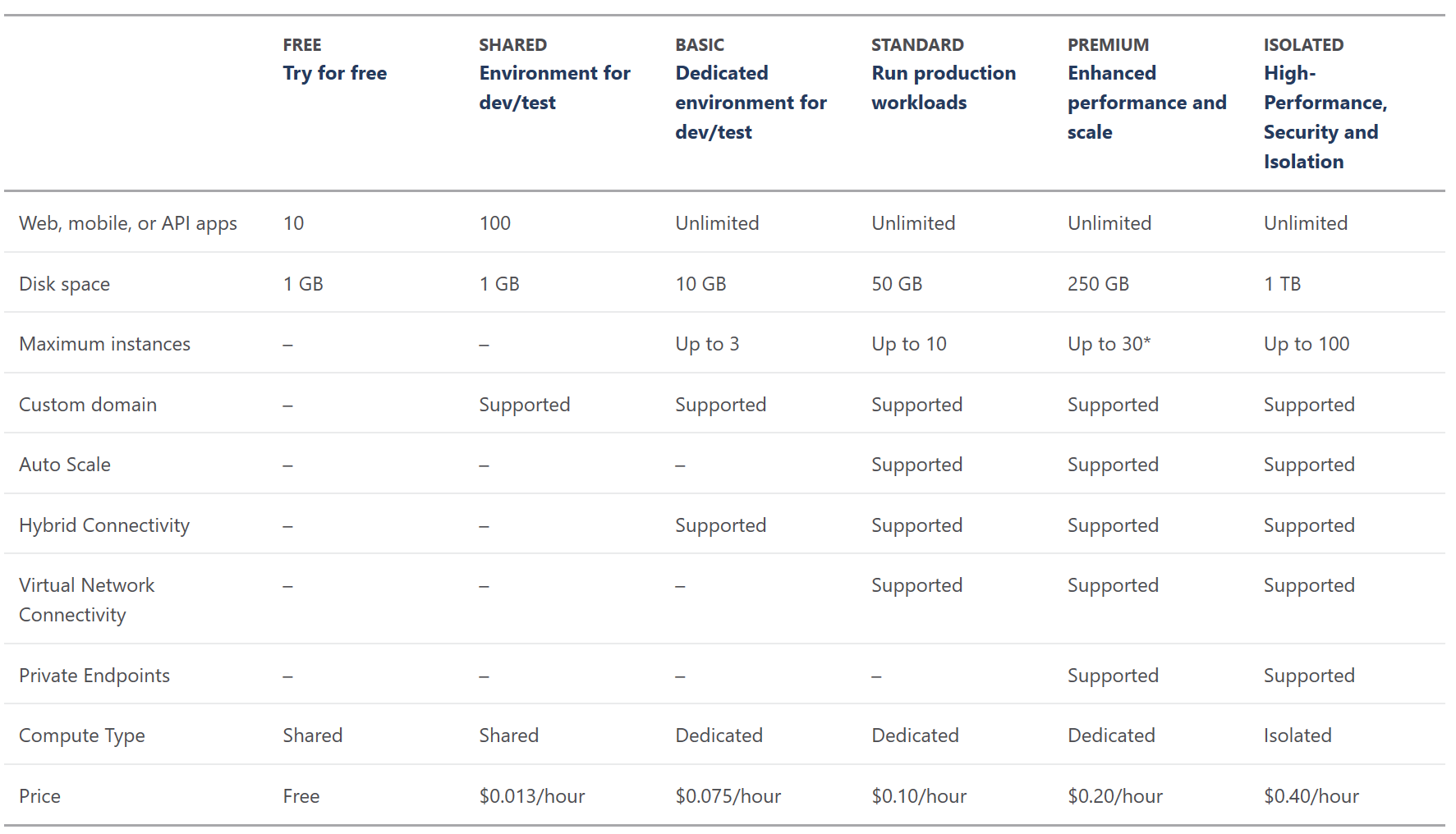
Description automatically generated

1. **Open Application in Browser**
2. Sandeep-webapp 🡪 Overview 🡪 Copy **Default domain** (sandeep-webapp.azurewebsites.net)
3. Browse[**https://sandeep-webapp.azurewebsites.net**](https://sandeep-webapp.azurewebsites.net)

**App Service Plan**

* App Service plans represent the collection of physical resources used to host your apps.
* App Service plans define:
  + Region (West US, East US, etc.)
  + SKU (Free, Shared, Basic, Standard, Premium, Isolated) – Features.
  + Instance size (Small, Medium, Large) – Memory and Processor
  + Scale count (one, two, three instances, etc.)
* Web Apps, Mobile Apps, API Apps, or Functions, in Azure App Service all run in an App Service plan. Apps in the same subscription, region, and resource group can share an App Service plan.
* All applications assigned to an **App Service plan** share the resources defined by it allowing you to save cost when hosting multiple apps in a single App Service plan.

Note: The requested app service plan cannot be created in the current resource group because it is hosting Linux apps. Please choose a different resource group or create a new one.



**Isolated**. This tier runs dedicated Azure VMs on **dedicated Azure Virtual Networks**, which provides network isolation on top of compute isolation to your apps. It provides the maximum scale-out capabilities.

Because a single resource group can have multiple App Service plans, you can allocate different apps to different physical resources that spans geographical regions. For example, a highly available app running in two regions includes at least two plans, one for each region, and one app associated with each plan. In such a situation, all the copies of the app are then contained in a single resource group. Having a resource group with multiple plans and multiple apps makes it easy to manage, control, and view the health of the application.

* It is recommended to isolate an app into a new App Service plan when:
* App is resource-intensive.
* App has different scaling factors from the other apps hosted in an existing plan.
* App needs resource in a different geographical region.
* You can move an app to a different App Service plan in the Azure portal. **You can move apps between plans as long as the plans are in the same resource group and geographical region.**

AS0, AS1, AS6 - AP1 (RG1/East US)

AS2 - AP2 (RG1/West US)

AS3 - AP3 (RG1/East India)

- AP4 (RG1/East US)

- AP5 (RG2/East US)

* You can create an **empty App Service plan** and then select the same while creating an App Service or you can create an App Service Plan while creating an App Service.
* If you want to move the app to a different region, one alternative is app cloning. Cloning makes a copy of your app in a new or existing App Service plan in any region. You can find **Clone App** in the **Development Tools** section of the menu. The web app must be running in the **Standard** mode in order for you to create a clone for the web app.

**Scaling an App Service Plan**

* Whether your application needs to handle a few hundred requests per day or a few million requests per day, the Azure Web Apps scalability features provide ways for you to deliver the right level of scale in a robust, cost-effective manner.
* When you consider the scalability requirements of an application, you should look at its resource requirements **vertically** (scaling up) **horizontally** (scaling out).
* You typically choose to **scale up** when any single request demands more memory and processing power to complete, and the ***bottleneck / latency in the system is the intensive number of software objects created in the computer’s memory or the intensive algorithms and business logic that is performed.*** When you scale up a web app, you increase the resource capacity, **such as RAM and CPU cores**, of the virtual machine on which your web app is running.
* You typically **scale out** when any single request requires **less** memory and processing power to complete, but the real ***bottleneck / latency is in network communication, disk access, etc.*** *In this case, the key to completing each request more efficiently is to* ***run it in parallel*** *to other requests as each wait on external components to complete****.*** To scale out a web app, you **increase the number of virtual machine/app service instances** on which your web app is running. For the properly architected app, this means your web app can handle more load and therefore service more user requests.

If you scale an app in the Basic tier to two instances, you have 350 concurrent connections for each of the two instances. For Standard tier and above, there are no theoretical limits to web sockets, but other factors can limit the number of web sockets. For example, maximum concurrent requests allowed (defined by maxConcurrentRequestsPerCpu) are: 7,500 per small VM, 15,000 per medium VM (7,500 x 2 cores), and 75,000 per large VM (18,750 x 4 cores).

**Scale Up (Vertical Scaling) the Azure Web App:**

* The ability to scale up a web app exists only for web apps configured for Basic, Standard, or Premium pricing tiers.
* The scale settings take only seconds to apply and affect all web apps in your App Service plan. They do not require your code to be changed or your applications to be redeployed.

Steps to vertically scale the application:

1. sandeep-webapp 🡪 Scale up (App Service Plan)
2. Select the Pricing tier based on following options:
   1. Number of Cores
   2. RAM
   3. Storage
   4. Slots (Number of CPU Instances)
   5. Backup frequency
   6. Traffic Manager facility

**To Scale Out: (Horizontal Scaling)**

**Auto scale based on CPU percentage:**

* The Target range setting defines the **minimum** and **maximum** CPU percentage to target.
* As long as the CPU percentage is within this range, Autoscale will not increase or decrease the number of instances.
* When the CPU percentage exceeds the maximum CPU percentage you specify, Autoscale will add an instance. If CPU percentage continues to exceed the maximum CPU specified, then Autoscale will add another instance.

At no point will you have more than the maximum number of instances specified in the Instances setting.

* Similarly, when CPU percentage falls below the minimum CPU percentage you specify, Autoscale will remove an instance. If CPU percentage continues to all below the minimum CPU percentage specified, then Autoscale will remove another instance. At no point will you have fewer than the minimum number of instances specified in the Instances setting

**Note**: The CPU percentage is measured as an **average across all instances**. For example, if you have two instances, one of which is running at 50 percent CPU and the other of which is running at 100 percent CPU, then the CPU percentage would be 75 percent for all the instances at that point in time

**Lab:**

1. Sandeep-webapp 🡪 Select **Scale Out** **(App Service Plan)** 🡪 Select Rule based 🡪 Click on Manage rules based scaling

A screenshot of a computer

Description automatically generated

1. Select autoscale 🡪 + Add a scale condition 🡪 Select as below 🡪 Add a rule

A screenshot of a computer

Description automatically generated

1. Wait till metric name is loaded 🡪 Enter details as below 🡪 Add

A screenshot of a graph

Description automatically generated

1. + Add a rule again 🡪 Enter details as below 🡪 Add

A screenshot of a graph

Description automatically generated

1. Repeat the same for MemoryPercentage
2. Save
3. Note that the Rules should appear as below

Text, email

Description automatically generated

**Deployment Slots**

* A deployment slot technically is an **independent** web app with its own content, configuration, and even a unique host name. So, it functions just like any other web app.
* Each Slot is reachable from its unique URL. For example for Staging deployment slot:

*http://dssdemoapp.azurewebsites.net/*

*http://dssdemoapp-****staging****.azurewebsites.net/*

* This option is available only in **Standard** and above pricing tier.

**Benefits of Deployment Slots:**

1. You can deploy changes for your application to a **staging deployment slot** and test the changes without impacting users who are accessing the **production deployment slot**. When you are ready to move the new features into production, you can just **swap the staging and production** slots **with no downtime** (**Blue-Green Deployment**)
2. You can **swap back** to the previous deployment if you realize that the new version of your application is not working as you expected.
3. You can "**warm up**" your application in a staging slot before swapping it into the production slot, avoiding the long delays a cold start of your application may incur because of some lengthy initialization code.
4. You can do **A/B testing** with a small set of users to try out new features of your application without impacting the majority of users who are using the production slot.

Note: A/B testing (also known as [**split testing**](https://www.optimizely.com/split-testing/) **or** [**bucket testing**](https://www.optimizely.com/optimization-glossary/bucket-testing/)) is a method of comparing two versions of a webpage or app against each other to determine which one performs better. AB testing uses data & statistics to validate new design changes and improve your conversion rates.

1. Can be used for **Canary** Deployments.

**Slot Supports basic on Plan**

Free / Shared / Basic = No slots support

Standard = 5 Slots

Premium = 20 Slots

**Lab: Adding a Deployment slot:**

1. App Services 🡪 Select App Service 🡪 Settings 🡪
   1. Scale Vertical 🡪 Select Standard S1
   2. Deployment Slots 🡪 + Add Slot (blade)
   3. Set Name=staging and Clone Configuration Settings of Production 🡪 Add

A screenshot of a computer

Description automatically generated

1. Visit Staging URL: <https://sandeep-webapp-staging.azurewebsites.net>

Format of **Domain Name** for Deployment Slot = <WebApp>-<**DeploymentSlotName**>.azurewebsite.net

1. Click and open slot: sandeep-webapp-staging
2. **Publish to Azure App Service Deployment Slot**
3. **Sandeep-webapp/staging** 🡪 **Deployment Center**
   * Source = Local Git
   * Save
   * Copy the Git Clone URL (<https://sandeep-webapp-staging.scm.azurewebsites.net:443/sandeep-webapp.git>)
   * Switch to Local Git/FTP credentials 🡪 Copy Local Git username and Password.

A screen shot of a screen

Description automatically generated

**Switch to your local machine:**

1. C:\demos\HelloWorldApp>git remote remove origin
2. C:\demos\HelloWorldApp>git remote add origin <slot url> eg: <https://sandeep-webapp-staging.scm.azurewebsites.net:443/sandeep-webapp.git>
3. C:\demos\HelloWorldApp>git push origin master

A screenshot of a computer

Description automatically generated

1. **Open Application in Browser**
2. Sandeep-webapp 🡪 Overview 🡪 Copy **Default domain** (sandeep-webapp.azurewebsites.net)
3. Browse[**https://sandeep-webapp-staging.azurewebsites.net**](https://sandeep-webapp-staging.azurewebsites.net)
4. **To Swap with Production:**
5. Sandeep-webapp 🡪 Deployment slots 🡪 **Swap**
   1. Source = sandeep-webapp-staging
   2. Destination = sandeep-webapp
   3. Swap
6. Browse[**https://sandeep-webapp.azurewebsites.net**](https://sandeep-webapp.azurewebsites.net)

# **Configure a Custom Domain Name in Azure App Service**

**Step 1:** Reserve the domain name. There are many domain registrars to choose from eg: GoDaddy.com

**Step 2:** Create **DNS records** that map the domain to your Azure web app.

The Domain Name System (DNS) uses data records to map domain names into IP addresses. There are several types of DNS records. For web apps, you’ll create either an ***A* record** or a ***CNAME*** **record**.

1. An A **(Address)** record maps a domain name to an IP address.
2. A **CNAME (Canonical Name)** 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. For example, you could map contoso.com to *<yourwebapp>*.azurewebsites.net.

www.dssdemoapp.com

1. Create CName record (**www.**dssdemoapp.com => dssdemoapp.azurewebsites.net)
2. Create Txt Record (asuid.www => 6D7F5E02604A7CC6A9FD19B8022AC8DD505ED4D51ADE0AB3D37B31ED639214C4)

dssdemoapp.com

1. Create a ARecord (@ => IP of Service ) (Only if we want to map Main domain eg: dssdemoapp.com)
2. Create Txt Record (asuid => 6D7F5E02604A7CC6A9FD19B8022AC8DD505ED4D51ADE0AB3D37B31ED639214C4)

**Note:** If the IP address changes, a CNAME entry is still valid, whereas an A record must be updated. However, some domain registrars do not allow CNAME records for the root domain or for wildcard domains. In that case, you must use an A record.

**Step 3:** Add the domain name inside the Azure Portal.

1. App Services 🡪 Select App Service 🡪 Settings 🡪 **Custom domains 🡪 Add Custom Domain**
2. Use the **DOMAIN NAMES** text boxes to enter the domain names to associate with this web app.

**Backup and Restore your app in Azure**

Snapshots automatically create periodic restore points of your app when hosted in a Standard or Premium App Service plan.

The Backup and Restore feature in Azure App Service lets you easily create app backups **manually** or on a **schedule**. You can restore the app to a snapshot of a previous state by overwriting the existing app or restoring to another app.

App Service can back up the following information to an Azure storage account and container that you have configured your app to use.

* App configuration
* File content
* Database connected to your app

Backups can be up to 10 GB of app and database content (max 4GB). If the backup size exceeds this limit**, you get an error**.

1. Sandeep-webapp 🡪 Backups
2. Configure the Backup
   * Backup Storage
   * Backup Schedule
   * Backup Database
3. Backup page 🡪 Click Backup Button
4. You see a progress message during the backup process.

**Optional: Integrate an App with an Azure Virtual Network**

VNet Integration enables apps to access resources in or through a VNet.

**Azure App Service has two variations on vNet Integration:**

* Multitenant systems that support the full range of pricing plans except Isolated.
* **App Service Environment (ASE),** which deploys into your VNet and supports **Isolated pricing plan** apps and **doesn't** require use of the VNet Integration feature.

VNet Integration gives your app access to resources in a VNet, but it doesn't grant inbound private access to your app from the VNet.

VNet Integration is used only to **make outbound calls** from your app into your VNet. The VNet Integration feature behaves differently when it's used with VNet in the same region and with VNet in other regions.

1. **Regional VNet Integration**: When you connect to Azure Resource Manager virtual networks **in the same region**, you must have a dedicated subnet in the VNet you're integrating with.
2. **Gateway-required VNet Integration**: When you connect to VNet in **other regions** or to a **classic virtual network** in the same region, you need an **Azure Virtual Network Gateway with Point to Site** provisioned in the target VNet.

Azure App Service 🡪 Networking 🡪 VNet Integration 🡪 Click here to configure 🡪 + Add VNet

**Using VNet Integration enables your app to access:**

* Resources in a VNet in the same region as your app. Private IP address of resources in VNet of can be used with programming code in App Service.
* Resources in VNets peered to the VNet your app is integrated with.
* Service endpoint secured services.
* Resources across Azure ExpressRoute connections.