Azure Function App: Azure Functions is a serverless solution that allows you to write less code, maintain less infrastructure, and save on costs. Instead of worrying about deploying and maintaining servers, the cloud infrastructure provides all the up-to-date resources needed to keep your applications running.

You focus on the code that matters most to you, in the most productive language for you, and Azure Functions handles the rest.

For the best experience with the Functions documentation, choose your preferred development language from the list of native Functions languages at the top of the article.

Scenarios

Functions provides a comprehensive set of event-driven triggers and bindings that connect your functions to other services without having to write extra code.

The following are a common, *but by no means exhaustive*, set of integrated scenarios that feature Functions.

If you want to...	then...
Process file uploads	Run code when a file is uploaded or changed in blob storage.
Process data in real time	Capture and transform data from event and IoT source streams on the way to storage.
Infer on data models	Pull text from a queue and present it to various AI services for analysis and classification.
Run scheduled task	Execute data clean-up code on pre-defined timed intervals.
Build a scalable web API	Implement a set of REST endpoints for your web applications using HTTP triggers.
Build a serverless workflow	Create an event-driven workflow from a series of functions using Durable Functions.
Respond to database changes	Run custom logic when a document is created or updated in Azure Cosmos DB.
Create reliable message systems	Process message queues using Queue Storage, Service Bus, or Event Hubs.

These scenarios allow you to build event-driven systems using modern architectural patterns. For more information, see Azure Functions Scenarios.

Development lifecycle

With Functions, you write your function code in your preferred language using your favorite development tools and then deploy your code to the Azure cloud. Functions provides native support for developing in C#, Java, JavaScript, PowerShell, Python, plus the ability to use more languages, such as Rust and Go.

Functions integrates directly with Visual Studio, Visual Studio Code, Maven, and other popular development tools to enable seamless debugging and deployments.

Functions also integrates with Azure Monitor and Azure Application Insights to provide comprehensive runtime telemetry and analysis of your functions in the cloud.

Hosting options

Functions provides a variety hosting options for your business needs and application workload. Event-driven scaling hosting options range from fully serverless, where you only pay for execution time (Consumption plan), to always warm instances kept ready for fastest response times (Premium plan).

When you have excess App Service hosting resources, you can host your functions in an existing App Service plan. This kind of Dedicated hosting plan is also a good choice when you need predictable scaling behaviors and costs from your functions.

If you want complete control over your functions runtime environment and dependencies, you can even deploy your functions in containers that you can fully customize. Your custom containers can be hosted by Functions, deployed as part of a microservices architecture in Azure Container Apps, or even self-hosted in Kubernetes.

```
resource "azurerm_app_service_plan" "example" {
  name                = "azure-functions-test-service-plan"
  location            = azurerm_resource_group.example.location
  resource_group_name = azurerm_resource_group.example.name

  sku {
    tier = "Standard"
    size = "S1"
  }
}

resource "azurerm_function_app" "example" {
  name                       = "test-azure-functions"
  location                   = azurerm_resource_group.example.location
  resource_group_name        = azurerm_resource_group.example.name
  app_service_plan_id        = azurerm_app_service_plan.example.id
  storage_account_name       = azurerm_storage_account.example.name
  storage_account_access_key = azurerm_storage_account.example.primary_
}
```