Microsoft

AZ-203.2 Module 04: Implement Azure Functions

Kishore Chowdary

Microsoft

Topics

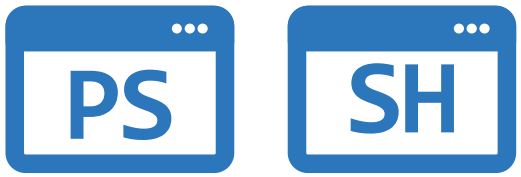
- Azure Functions
- Develop Azure Functions by using Visual Studio

Lesson 01: Azure Functions

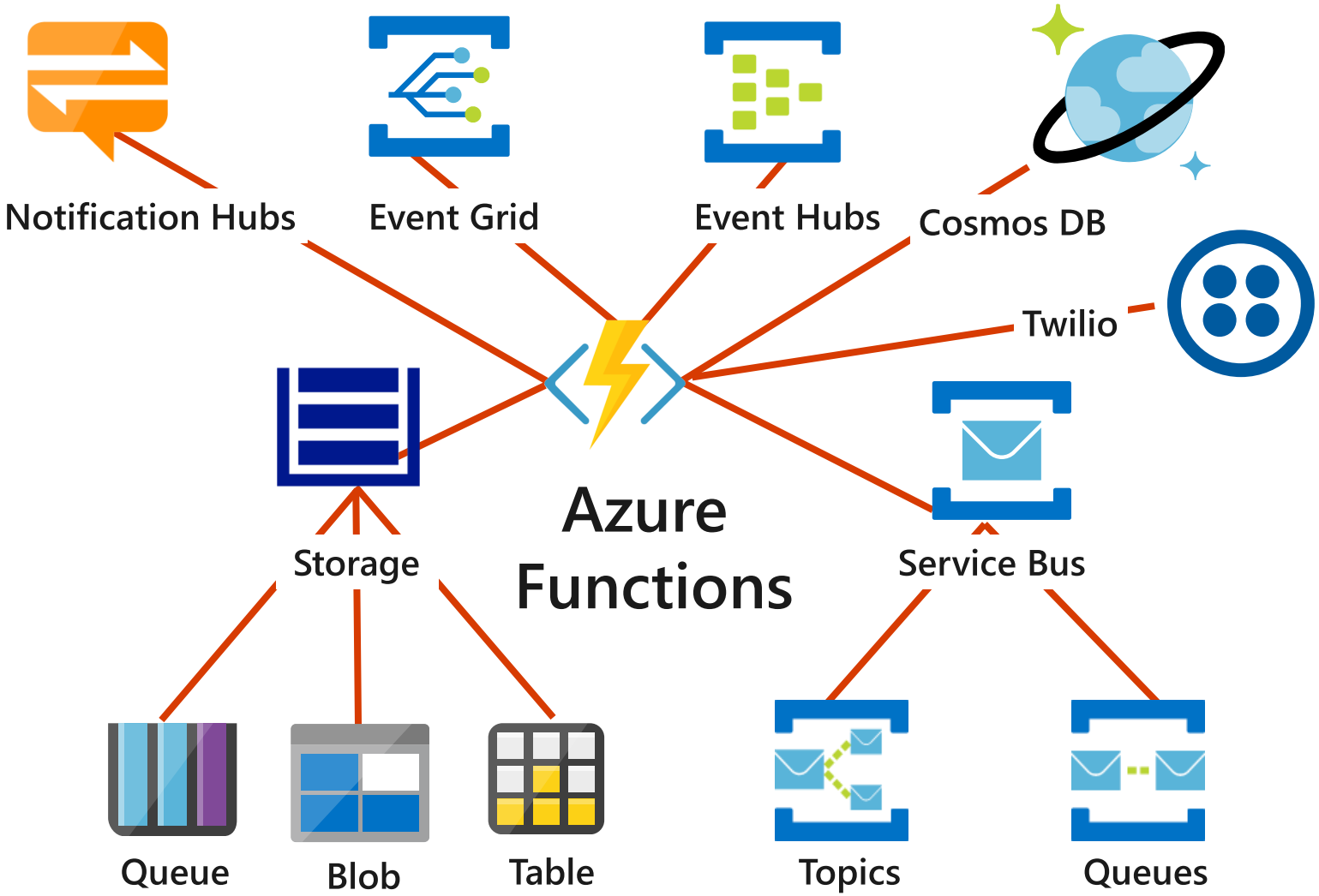
Azure Functions

- Solution for running small pieces of code, or "functions," in the cloud:
  - Write only code that is relevant to business logic
  - Removes the necessity to write “plumbing” code to connect or host application components
- Build on open-source WebJobs code
- Supports a wide variety of programming languages, for instance:

- Even supports scripting languages, such as:



Function integrations



Azure Function (Java program – Function.java)

```
public class Function {
    public String echo(
        @HttpTrigger(
            name = "request",
                methods = {"post"},
            authLevel = AuthorizationLevel.ANONYMOUS
        )
        String request, ExecutionContext context) {
        return String.format(request);
    }
}
```

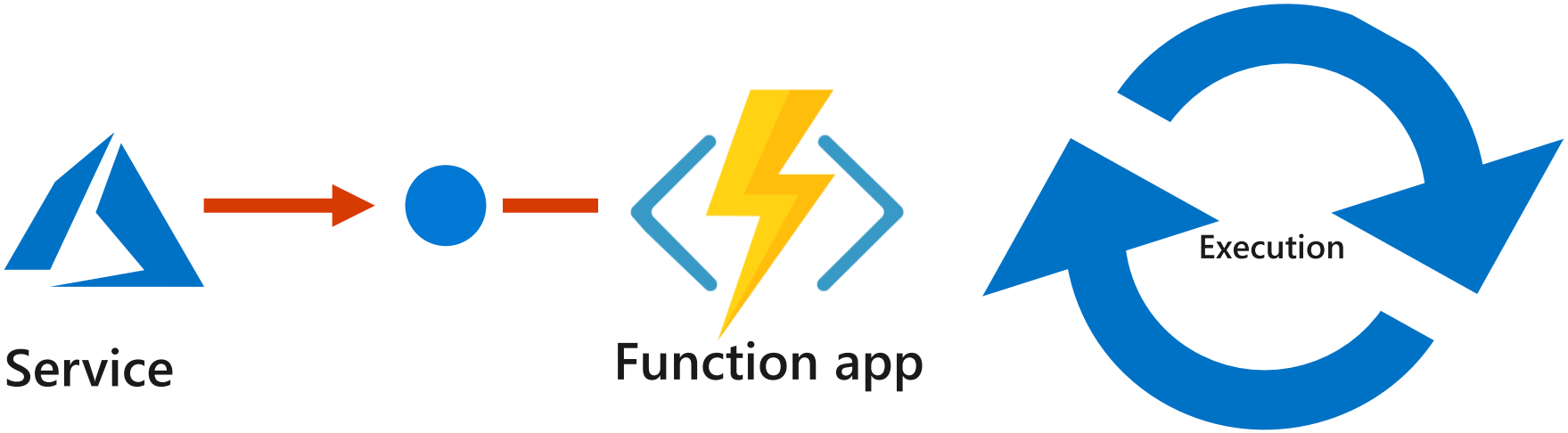
Azure Function (Python script – __init__.py)

```python
import logging

import azure.functions as func

def main(myblob: func.InputStream):
    logging.info(f"Python blob trigger function processed\n"
        f"Name: {myblob.name}\n"
        f"Blob Size: {myblob.length} bytes")
```

Scale and hosting

- You can choose between two types of plans
  - Consumption
    - Instances are dynamically instanced and you are charged based on compute time
  - App Service plan
    - Traditional App Services model used with Web Apps, API Apps, and Mobile Apps
- The type of plan controls:
  - How host instances are scaled out
  - The resources that are available to each host

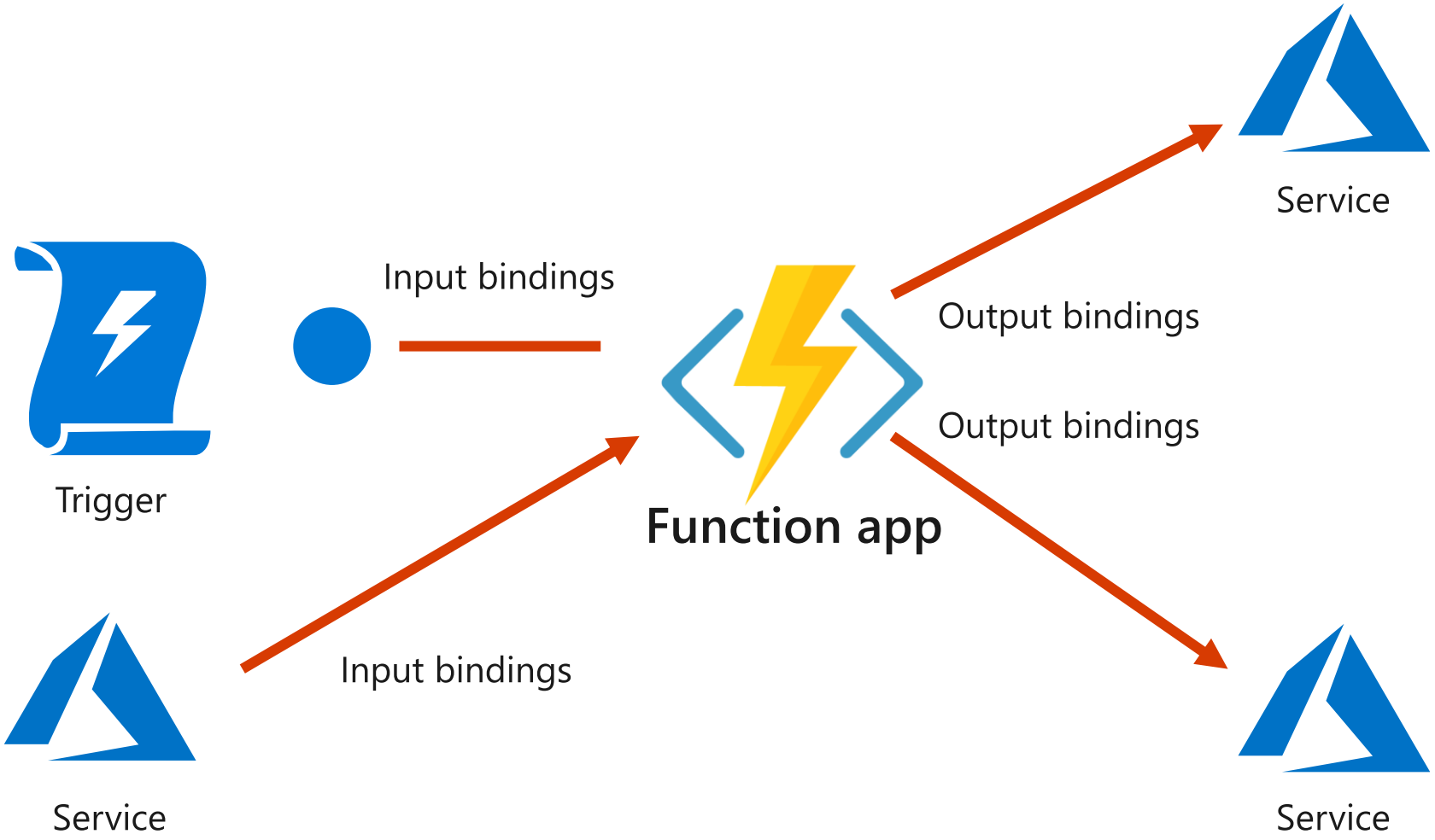
Triggers



Trigger types

- Triggers based on Azure services
  - Cosmos DB
  - Blob and queues
  - Service Bus
  - Event Hub
- Triggers based on common scenarios
  - HTTP request
  - Scheduled timer
- Triggers based on third-party services
  - GitHub
- And more...

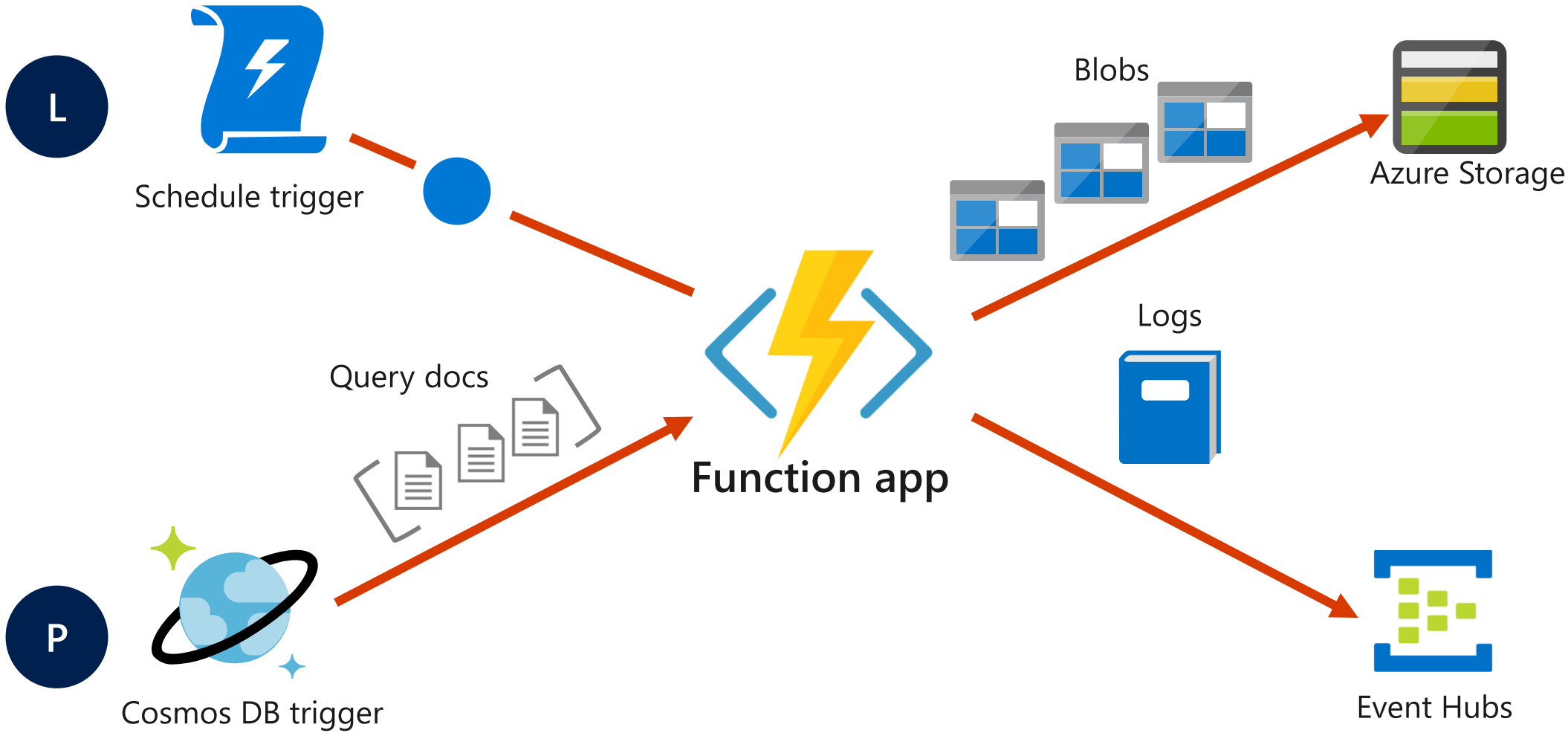
Input and Output Bindings



Bindings

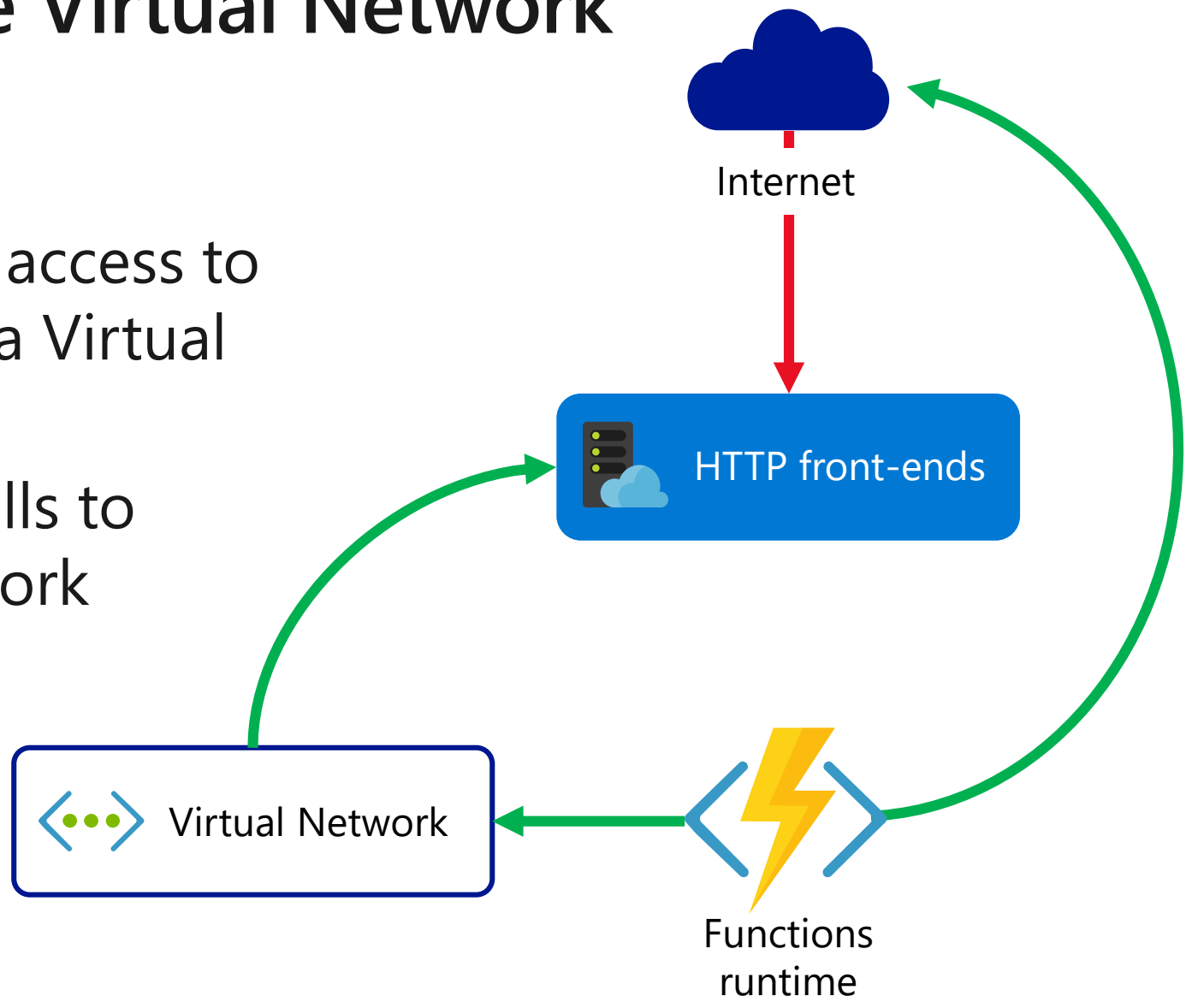
- Declarative way to connect to data from your code
  - Connect to services without writing plumbing code
  - Service credentials are not stored in code
  - Bindings are optional
- Function can have multiple input and output bindings
- Output bindings can send data to Azure services such as
  - Storage
  - Azure Cosmos DB
  - Service Bus

Trigger and Bindings example

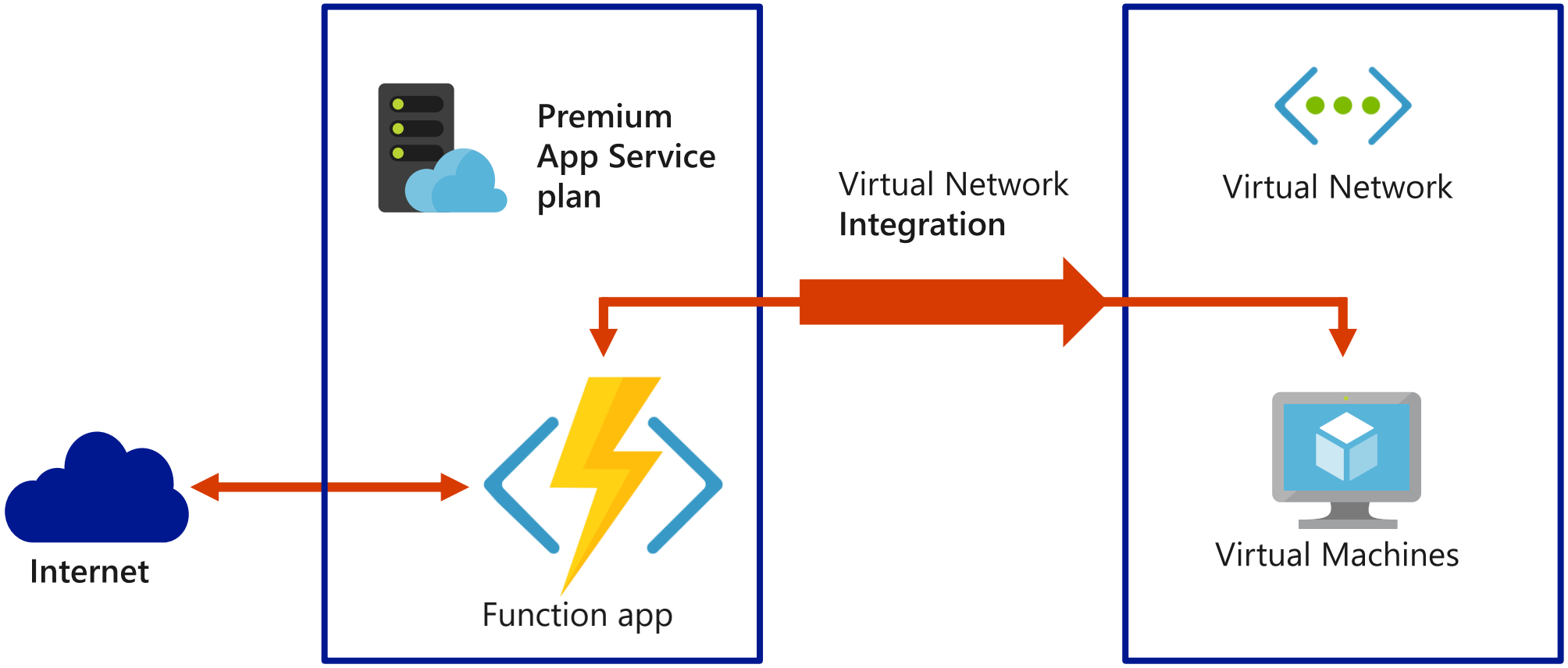


Integrating with Azure Virtual Network

- Requires the Premium plan
- Secures the inbound HTTP access to your app to one subnet in a Virtual Network
- Allows secure outbound calls to resources in a Virtual Network



Azure Virtual Network integration example



Best practices

- Avoid long-running functions
  - Functions that run for a long time can time out
- Use queues for cross-function communication
  - If you require direct communication, consider Durable Functions or Azure Logic Apps
- Write stateless functions
  - Functions should be stateless and idempotent
  - State data should be associated with your input and output payloads
- Code defensively
  - Assume that your function might need to continue from a previous fail point

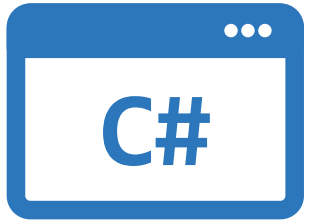
Lesson 02: Develop Azure Functions by using Visual Studio

Azure Functions in Visual Studio

- Visual Studio project type
  - Develop, test and deploy C# functions to Azure
- Use WebJobs attributes to configure functions in C# code
- Pre-compile C# functions
  - Better cold-start performance

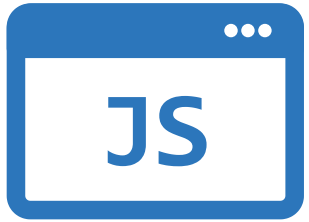
Function code

```
using System;
using Microsoft.Azure.WebJobs;
using Microsoft.Azure.WebJobs.Host;

namespace FunctionApp1
{
    public static class Function1
    {
        [FunctionName("QueueTriggerCSharp")]
        public static void Run([QueueTrigger("myqueue-items", Connection =
"QueueStorage")]string myQueueItem, TraceWriter log)
        {
            log.Info($"C# Queue trigger function processed: {myQueueItem}");
        }
    }
}
```



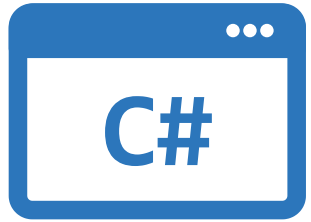
Bindings

```
{
    "bindings": [
        {
            "name": "order",
            "type": "queueTrigger",
            "direction": "in",
            "queueName": "myqueue-items",
            "connection": "MY_STORAGE_ACCT_APP_SETTING"
        },
        {
            "name": "$return",
            "type": "table",
            "direction": "out",
            "tableName": "outTable",
            "connection": "MY_TABLE_STORAGE_ACCT_APP_SETTING"
        }
    ]
}
```



Binding-based code

```
#r "Newtonsoft.Json"

using Microsoft.Extensions.Logging;
using Newtonsoft.Json.Linq;

public static Person Run(JObject order, ILogger log)
{
    return new Person() {
        PartitionKey = "Orders",
        RowKey = Guid.NewGuid().ToString(),
        Name = order["Name"].ToString(),
        MobileNumber = order["MobileNumber"].ToString()
    };
}
```



Demo: Creating an Azure Functions project