Azure Functions

CSCI E-94

Fundamentals of Cloud Computing - Azure

Joseph Ficara

Agenda

- Azure Functions
  - Overview
  - Serverless API
  - TimerTrigger
  - BlobTrigger
  - QueueTrigger

Overview

- Azure functions:
  - "code" that run "Serverless"
  - Still tied to a service plan
    - New plan "Consumption"
    - Can still use traditional app service plan(s)
  - Based on WebJobs
  - Inexpensive when using Consumption plan
    - 1 million executions per month for FREE!
      - 400,000 GB-s
    - See Azure Calculator

Overview

- Pricing calculation
  - Executions * (Duration in seconds)
    - = Execution Duration (in seconds)
  - Resource consumption (GB-s)
    Unit of measure: Gigabyte seconds
    - Resource Consumption in GB * Execution Duration in Seconds
      - = Total GB-s

Overview

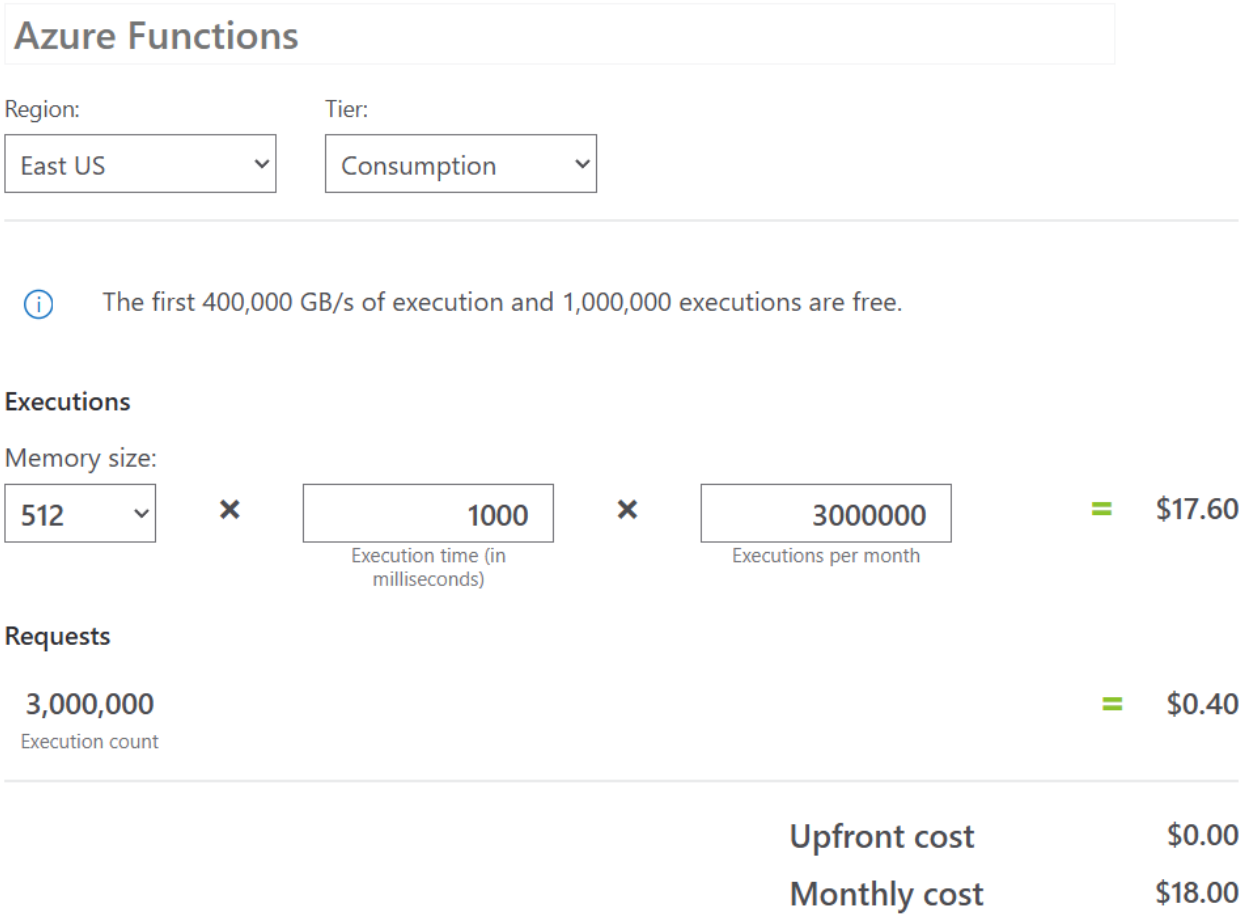
- Example:
  - Executions: 3 million
  - Duration: 1 second per execution
  - = 3 million seconds
  - 512 MB per execution (.5 GB)
  - Total GB-s = .5 * 3 million seconds
    - = 1.5 million GB-s

Overview

- What does it cost you?
  - 3 Million Executions
  - 1 Second Per Execution
  - 0.512 GB Per Execution
  - (3,000,000 * 1 * 0.512) = ~1.5 Million GB-s
  - - 400,000 GB-s (Monthly free grant)
  - (1.5 Million GB-s – 400,000 GB-s) = 1.1 million GB-s
  - $0.000016/GB-s Resource consumption price per GB-s
  - ($0.000016GB-s * 1.1 million GB-s) = $17.60
  - Execution cost = 0.20 per million, 1st million is free
    - Execution cost: 3 Million - 1 Million = 2 Million = 2 * 0.20 = 0.40
  - Resulting cost is $17.60 + 0.40 = ~$18.00

Overview

- What does it cost you? - Azure Calculator



Overview

- Why do you care?
  - Inexpensive
    - Pay per use model
  - Push you towards single responsibility
  - Formal support for many languages
  - Support for Durable functions
    - Stateful functions
  - Open-source http://bit.ly/2Td2ii7

Overview

Framework & Language Support Matrix

Language	1.x EOL 09/14/2026	2.x EOL 12/13/2022	3.X EOL 12/13/2022	4.x Current
C#	GA (.NET Framework 4.8)	GA (.NET Core 2.1)	GA (.NET Core 3.1)	GA (.NET 8) GA (.NET 9) GA (.NET Framework 4.8.1)
JavaScript Node.js TypeScript	GA (Node 6)	GA (Node 10 & 8)	GA (Node 14,12, & 10)	GA (18, 20) 22 Preview
F#	GA (.NET 4.8)	GA (.NET Core 2.1)	GA (.NET Core 3.1)	GA (.NET 8) GA (.NET 9) GA (.NET Framework 4.8.1)
Java	N/A	GA (Java 8)	GA (Java 11 & 8)	GA (Java 8, 11, 17) (Java 21 Linux only)
PowerShell	N/A	N/A	N/A	GA (7.4) 7.2 EOL November 8, 2024
Python	N/A	GA (Python 3.7)	GA (Python 3.8,3.7,3.6)	*GA (3.11, 3.10, 3.9)* *3.8 EOL October 2024*
Go/Rust/Other	N/.A	Linux/Windows Custom Handlers	*Linux/Windows* Custom Handlers	*Linux/Windows* Custom Handlers

See: http://bit.ly/3cnsC0C

Overview

- **Consumption** plan notes
  - See: http://bit.ly/3bRgkOK
  - Function can run for a max of 10 minutes
    - The **default timeout is 5 minutes**
      - Set in host.json via functionTimeout
  - When triggering you may see a delay
    - This **can be up to 10 minutes**
    - If faster triggering is needed
      - Switch to a traditional app service plan
      - Ensure its set to always on

Overview

- App Service Plans are also supported
  - Triggering is faster
  - Can be set to unlimited execution time
    - Note on the **premium plan**
      - Functions 2.0 - 4.0 default is **30 minutes**
        - **Functions 2.0 - 4.0 support never timeout (unlimited)**
- Containers
  - Working with containers & Azure Functions
- Kubernetes
  - Azure Functions on Kubernetes with KEDA

Overview

- HTTP Triggered function must respond
  - 230 seconds Windows / 240 on Linux
    Application performance FAQs for Web Apps in Azure
    - Azure Load Balancer timeout dependent
      - Options:
        - Use Durable Functions Async pattern
        - Defer actual work and return immediately

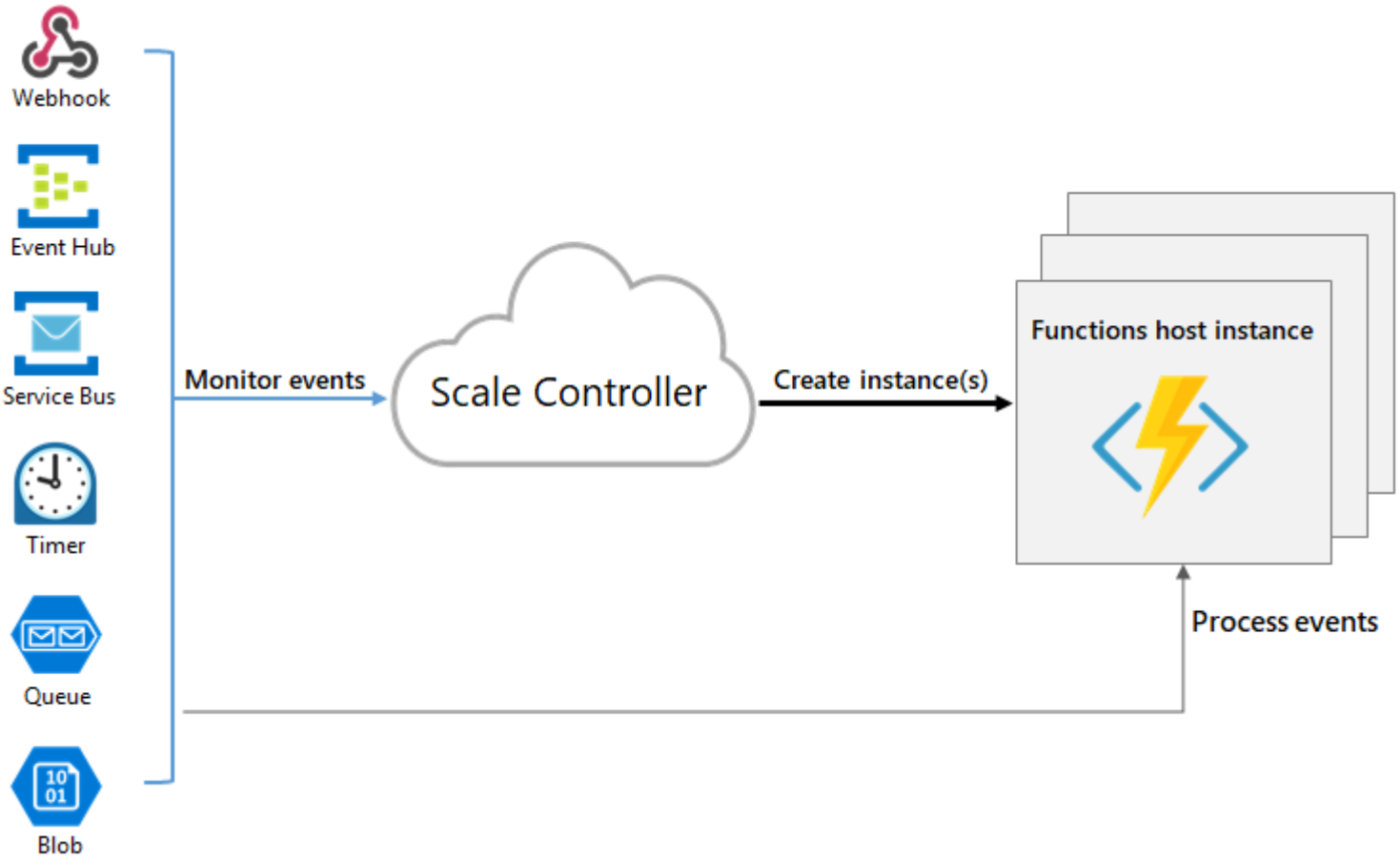
Overview

- Azure functions require
  - General Azure Storage Account
    - Supports Azure Blob, Queue, Files and Table
  - Some storage accounts don't support
    - Queues & Tables
  - Always On to have fast triggering support
    - This requires a traditional app service plan

Overview

- Scaling Azure Functions hosting options
  Event-driven scaling in Azure Functions
  - Is automatic with Consumption Plan
  - Uses the "Scale Controller"
    - To monitor rate of events and scale out or in
      - **SCALE_CONTROLLER_LOGGING_ENABLED**
        Scale controller logs
  - Max of 200 instances
    - Single function can handle more than one request
  - New Instance Rate
    - HTTP Triggers: Max once per second
    - Non-HTTP Triggers: Max once every 30 seconds
      - Scaling is faster in the premium plan

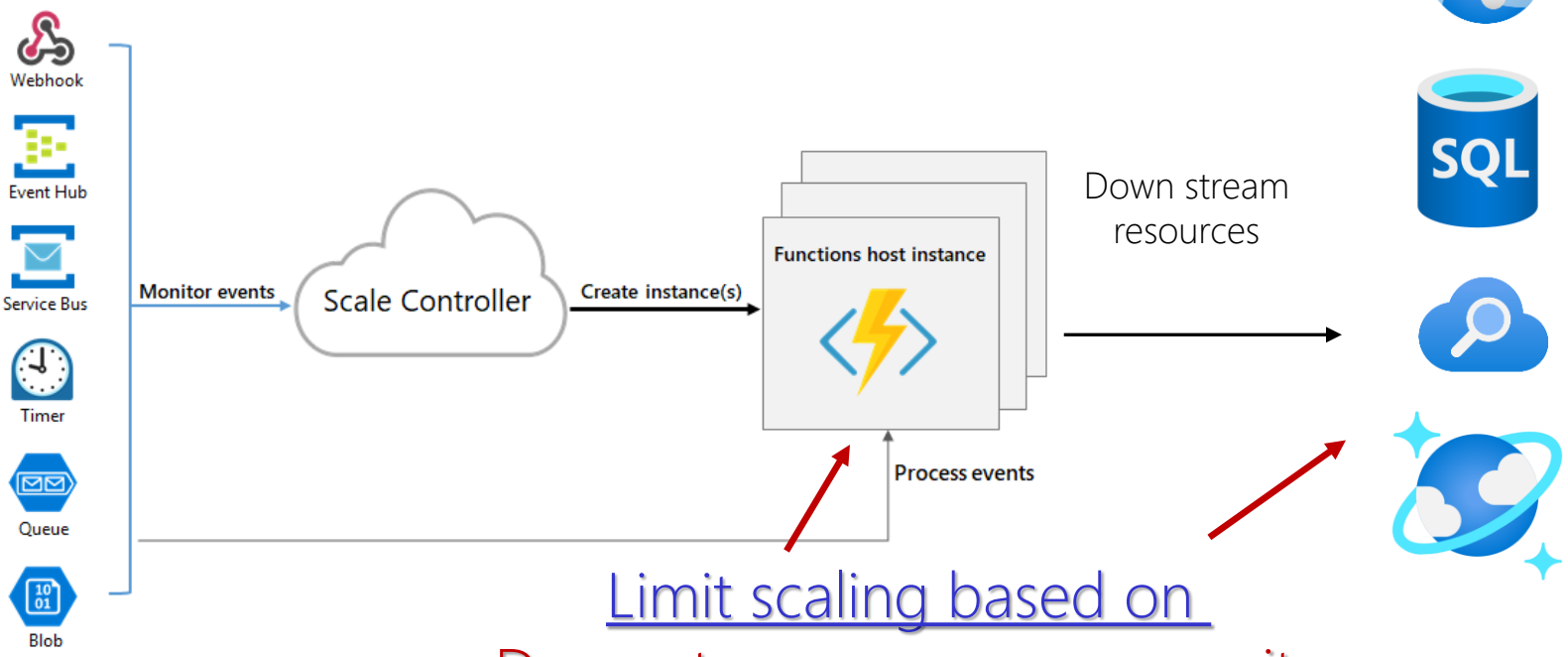
Overview
Scaling …



Courtesy Microsoft: Event-driven scaling in Azure Functions

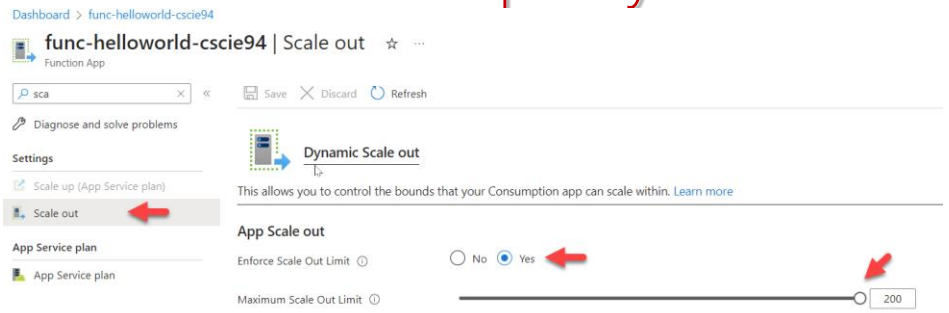
Overview

Scaling …



Limit scaling based on

Down stream resource capacity



Overview

- Azure functions support triggers
  - Triggers cause function execution
    - Blob Storage
    - Cosmos DB
    - Event Grid
    - Event Hub
    - HTTP
    - SignalR Service
    - Microsoft Graph Events
    - Queue storage
    - Service bus
    - Timer
    - Webhooks
    - Many more see Trigger and binding definitions

Overview

- Azure functions input/output binding
  - Binding causes retrieval and “storage”
    - Configured in function.json
  - Input / Output:
    - Blob Storage
    - Cosmos DB
    - Event Hubs
    - Microsoft Graph Events
    - And many more
      - See: Azure Functions triggers and bindings concepts

Overview

- Durable Functions & Logic Apps
  - Manage state transition between functions
  - Another option
    - Storage queues
- Event hubs are suited for
  - High volume communications

Overview

- Best practices
  - Do
    - Use async methods
    - Share HTTP Client instances
    - Receive messages in batches when possible
    - Configure host to better handle concurrency
      - host.json
    - Make functions stateless
    - Write defensive functions
      - Handle poison queue messages

Overview

- Best practices
  - Don't
    - Use blocking calls
    - Create long running functions

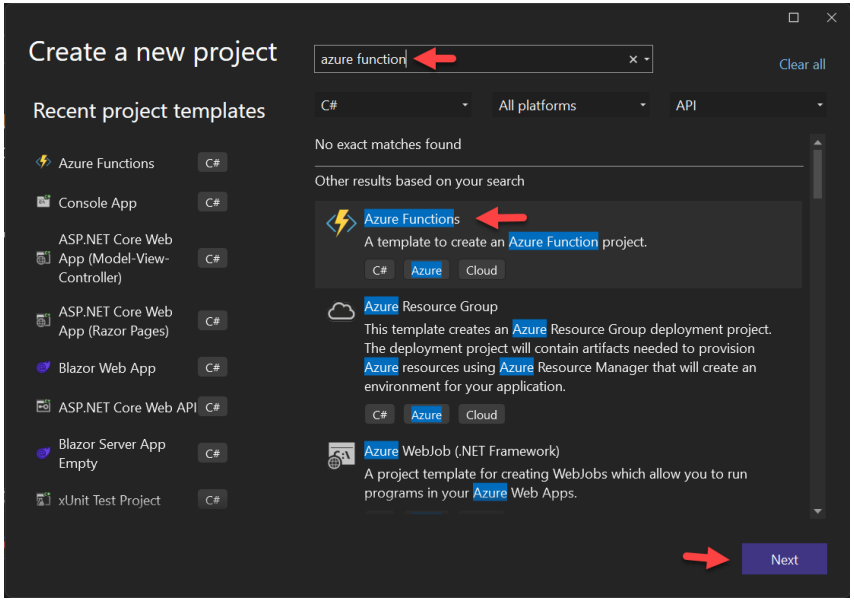
Questions ?

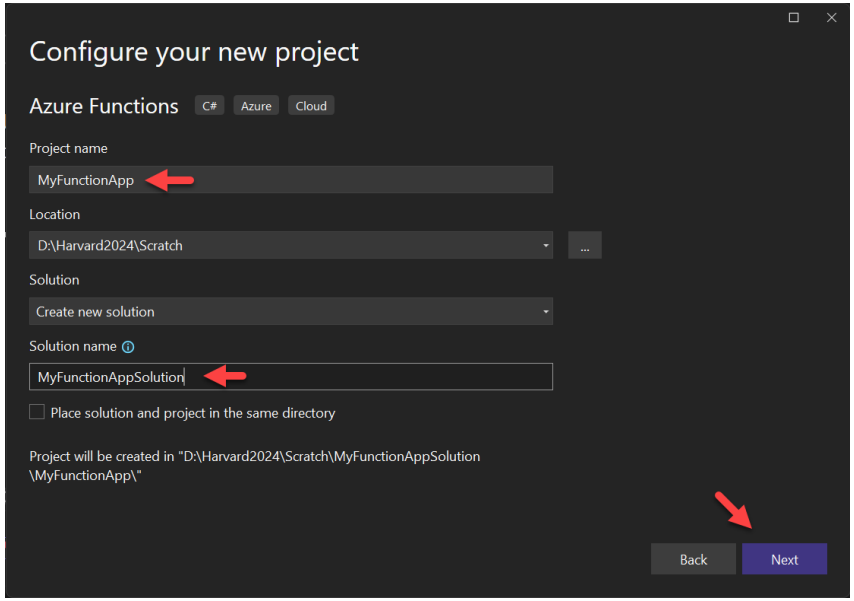
Hello World

- Let's look at some code…
- Simple Azure Function
  - Handles HTTP GET and POST
    - Using a single method
  - Returns "Hello <*inputdata*>"

Hello World

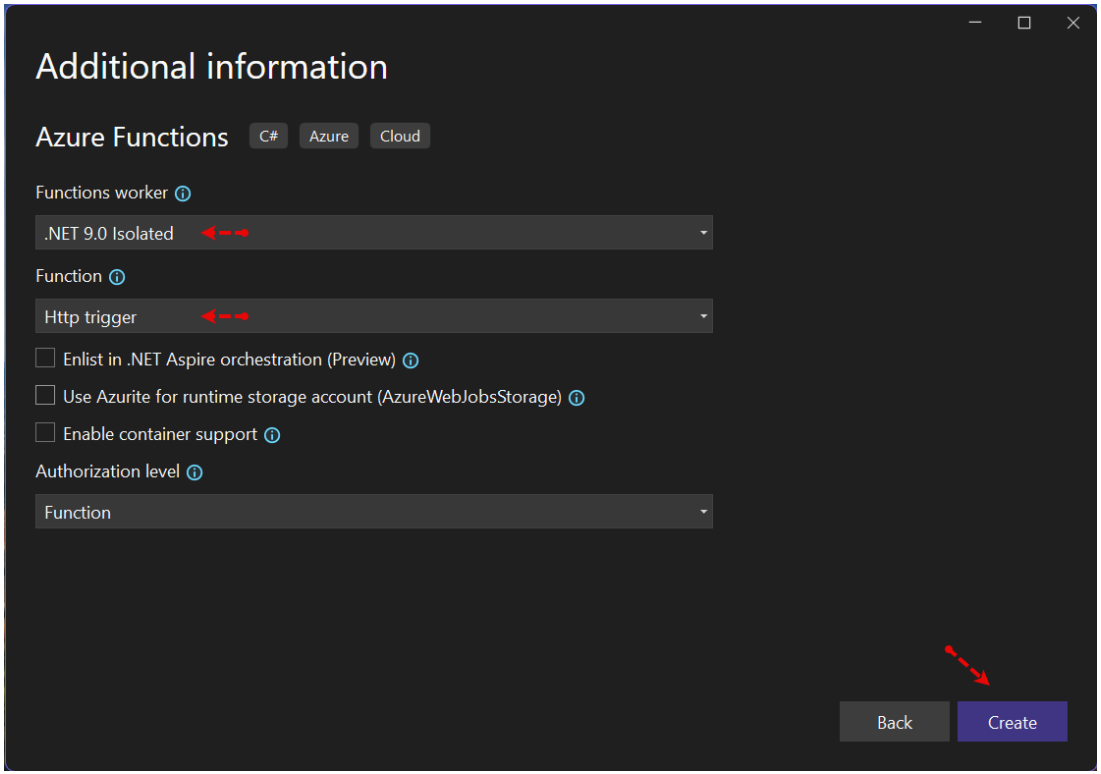
- To create an Azure Function project
  - Create a new project ->and type function





Hello World

- To create an Azure Function project
  - Http trigger and Function Authorization



Demo

Hello World Demo

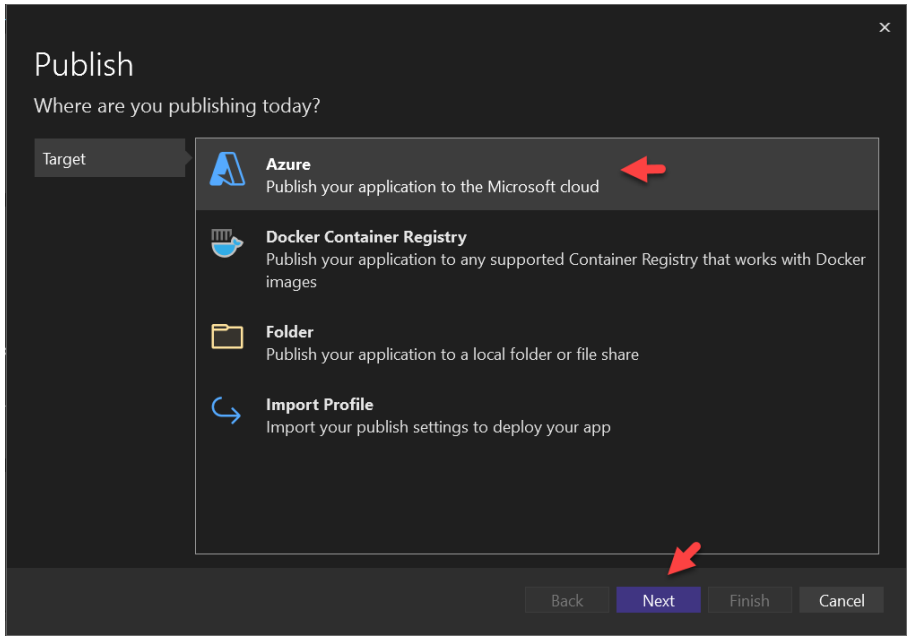
`MyFunctionAppSolution`

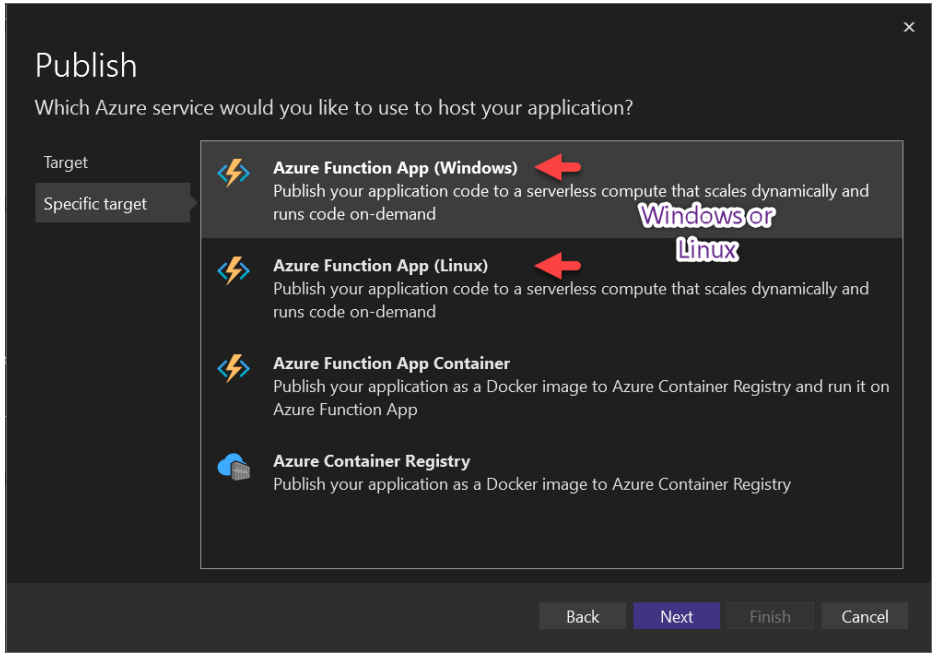
`HelloWorldAzureFunctionSolution`

Deploying an Azure Function

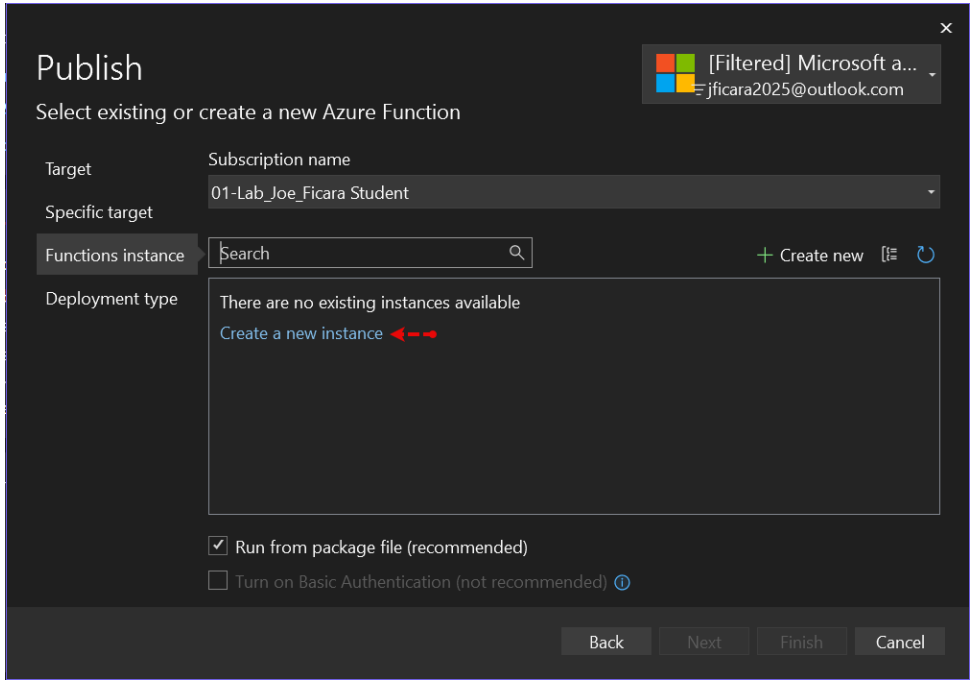
- Deployment
  - Consumption plan
  - Premium plan
  - App service plan Windows / Linux
- Create via
  - Visual Studio
  - Portal

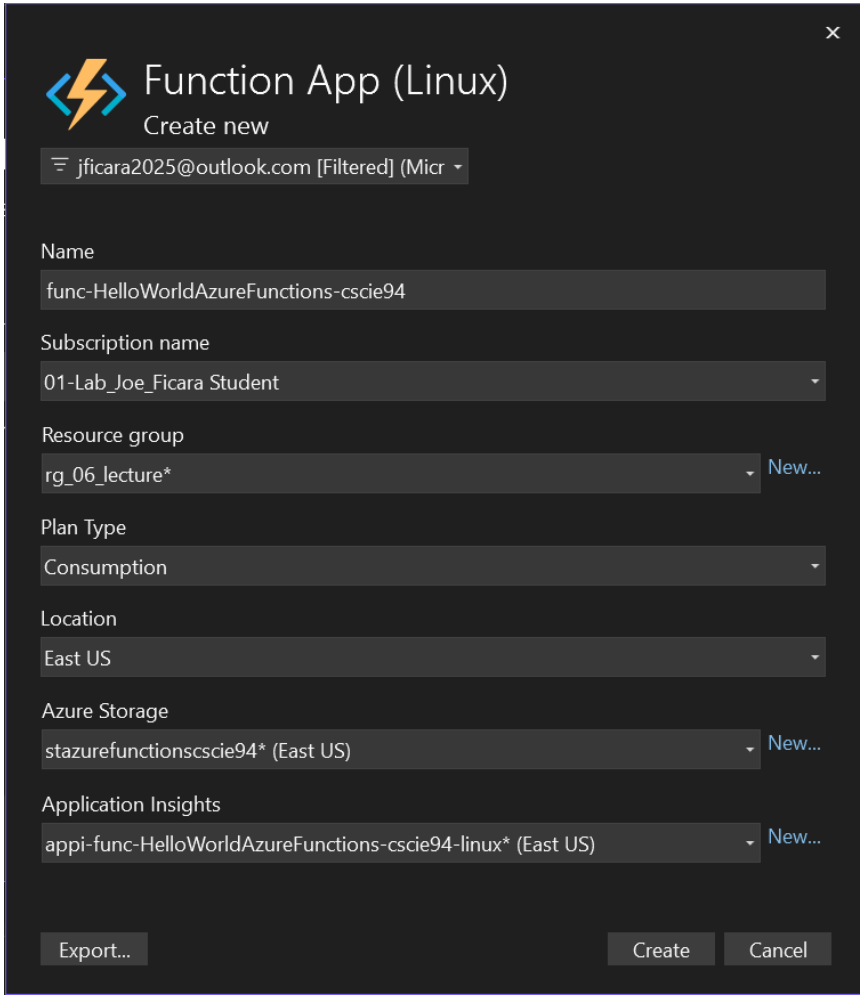
Deploying an Azure Function





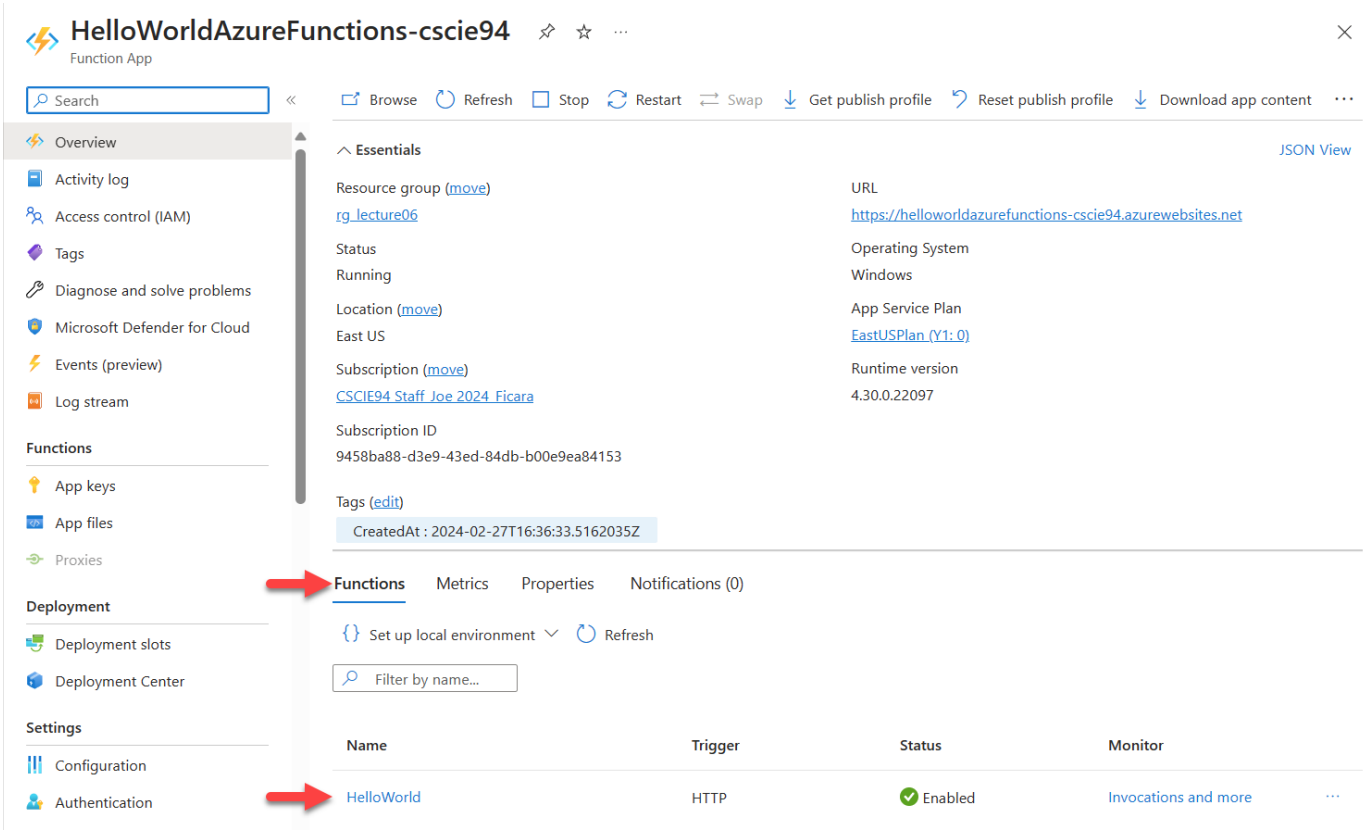
Deploying an Azure Function





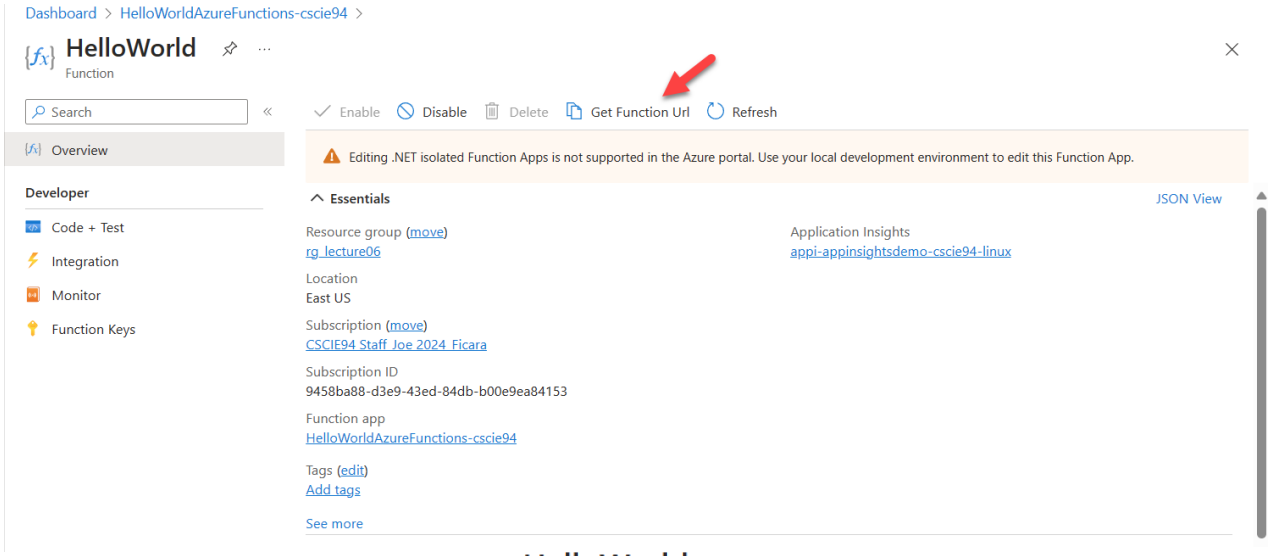
Deploying an Azure Function

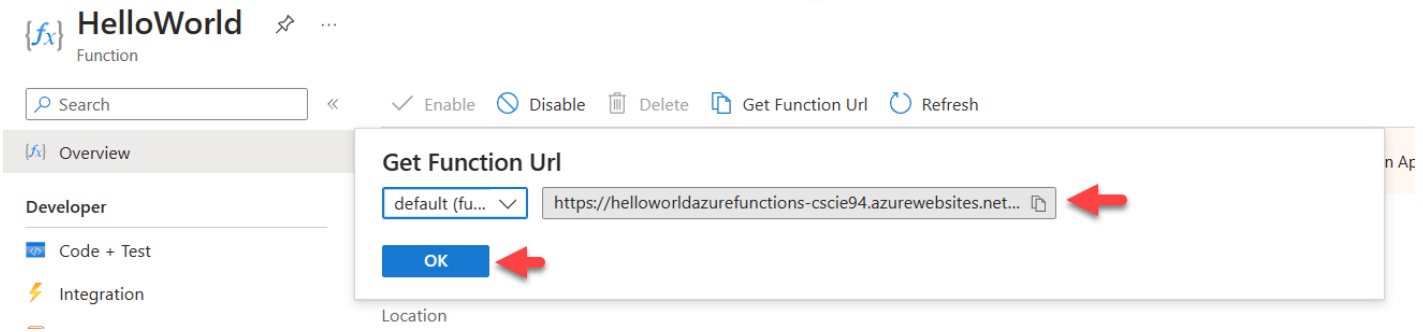
- To get URL to send requests to



Deploying an Azure Function

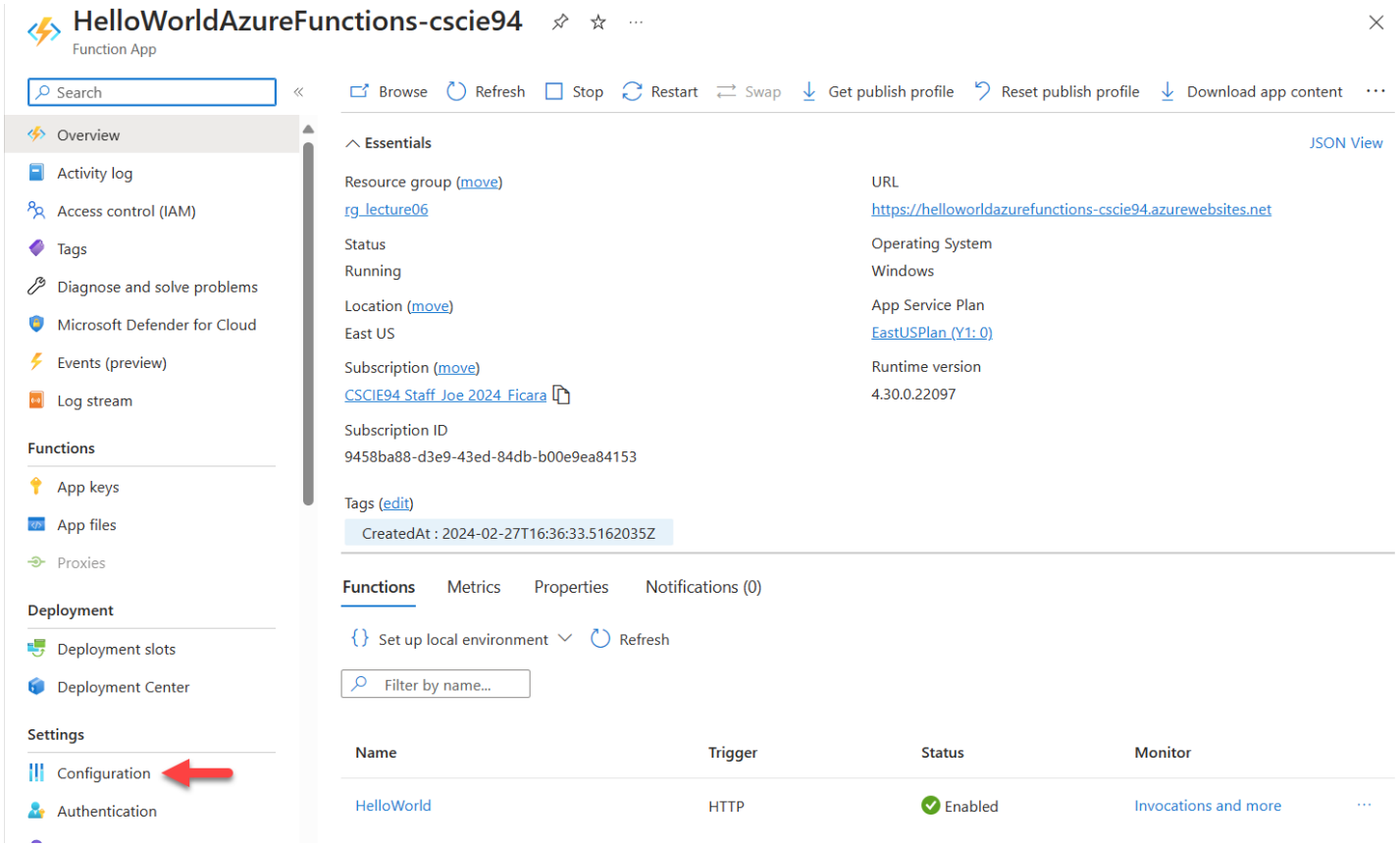
- To get URL to send requests to





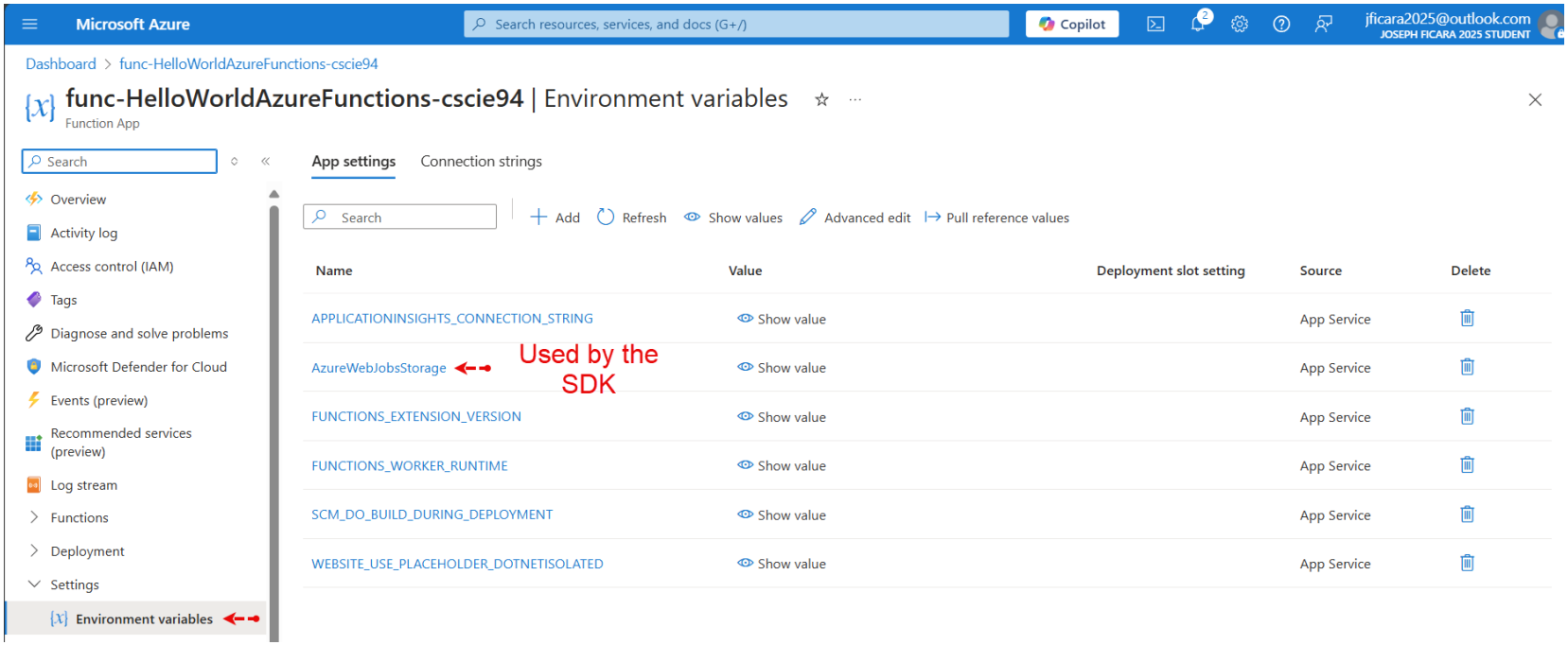
Deploying an Azure Function

- Blade now looks like standard App Service



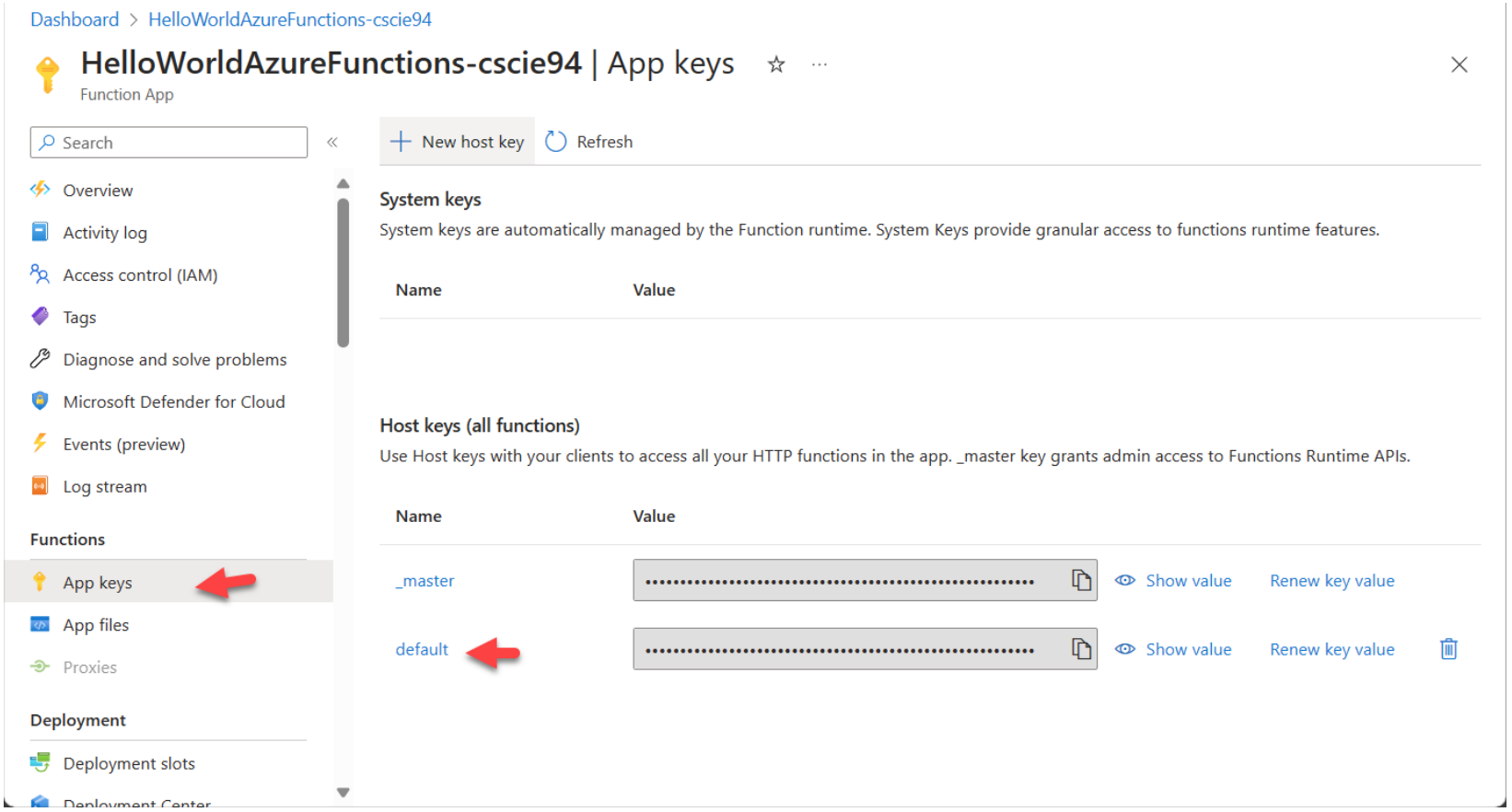
Deploying an Azure Function

- Storage configuration information
  - Via Environment Variables



Deploying an Azure Function

- Function Keys – defined under **App keys**



Deploying an Azure Function

- Two types of Keys
  - **Function** keys
    - **Specific to a function**
  - **Host** Keys
    - **All functions**
- 5 Authorization Levels
  - Anonymous
  - User
  - Function
  - System
  - Admin

Deploying an Azure Function
5 Authorization Levels

```
[FunctionName("Function1")]
0 references | Joseph Ficara, 2 days ago | 1 author, 1 change
public static async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Function, "get", "post", Route = null)] HttpRequest req, ILogger log)
```

- Anonymous
  - No authentication
- **Function**
  - Scope is a function – Is a Function key
- Admin
  - Requires a host key
  - Scope all functions

Deploying an Azure Function
5 Authorization Levels

- System
  - Scope all functions
  - Requires a master key
    - Named _master
- User
  - User based authentication
    - Azure Active Directory
    - Microsoft
    - Facebook
    - Google
    - Twitter

Demo

Deployment

`HelloWorldAzureFunctionSolution`

Serverless API

- Azure functions support
  - HTTP triggers
    - Think
      - REST API
      - Webhook
  - Authentication
  - Multi-verb methods
    - HTTP POST, GET all on same method
      - I am not a fan..
        - Except perhaps webhooks

Serverless API

- Should you create REST APIs this way?
  - If standalone
    - For infrequent usages, its ok
    - For complex REST interfaces likely no
  - If with API Management
    - Way to create microservice REST implementations

Serverless API

- How do you do it?
  - Using a C# Attribute
    - HttpTrigger
    - Specify
      - Authorization Level
      - HTTP Verb(s)
      - Route

Serverless API

- Authorization Levels
  - Anonymous
    - No authentication
  - Function
    - Function Key based
  - Admin
    - Host Key Based

Serverless API

- Authorization Levels
  - System
    - Master Key Based (Special host key)
    - Can’t be revoked
    - Can be renewed so older value won’t be used

Serverless API

- Authorization Levels
  - User
    - User authentication
    - Token based
    - Supports:
      - Azure Active Directory
      - Facebook
      - Google
      - Twitter
      - Microsoft Account

Serverless API

- We are using NET Core
  - Routing and Constraint syntax
    - See: Routing in ASP.NET Core
- Data sent via the
  - URL can be accessed through
    - Parameters or the request's Query property
- Data sent via the request body
  - Retrieved via the request Content property

Serverless API

- Configuration is done in the host.json file
  - host.json reference for Azure Functions 2.x and later
    - routePrefix
      - Default is: **api**
    - maxOutstandingRequests
      - Default is: **200** (unbounded -1 for ver 1.x)
    - maxConcurrentRequests
      - Default is: **100** (unbounded -1 for ver 1.x)
    - dynamicThrottlesEnabled
      - Default is: **true** (false for ver 1.x)
        - Checks performance counters reject if to busy (429)
          - If > 80% threshold

Demo

HTTP Triggers

`ServerlessAPIAzureFunctionSolution`

Blob Trigger

- Blob triggers
  - Think Queue based on blobs
    - When a blob is added to a container
      - A function is Invoked
- Why do you care?
  - Simple way to process data
    - Without adding another “component”

Blob Trigger

- How do you set it up?
  - Use the BlobTrigger attribute
  - Specify
    - The container you want to monitor
    - The parameters you want
      - You can get the BlobClient object
      - The Stream of the blob uploaded
      - The name of the blob
  - Implement your function!

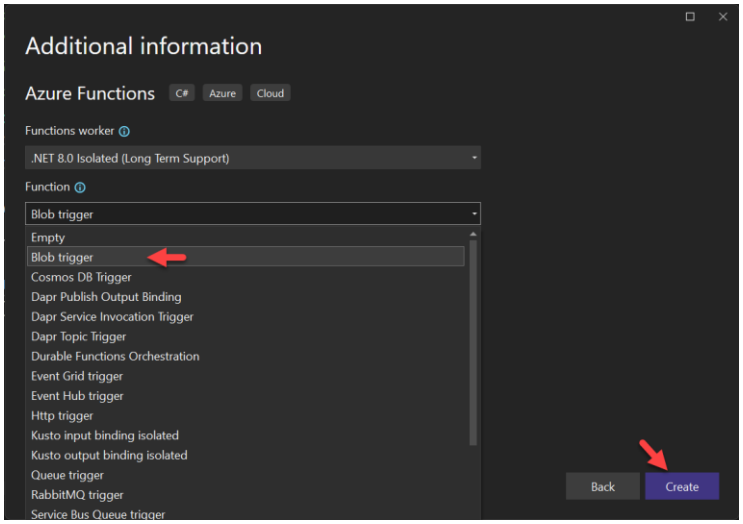
Blob Trigger

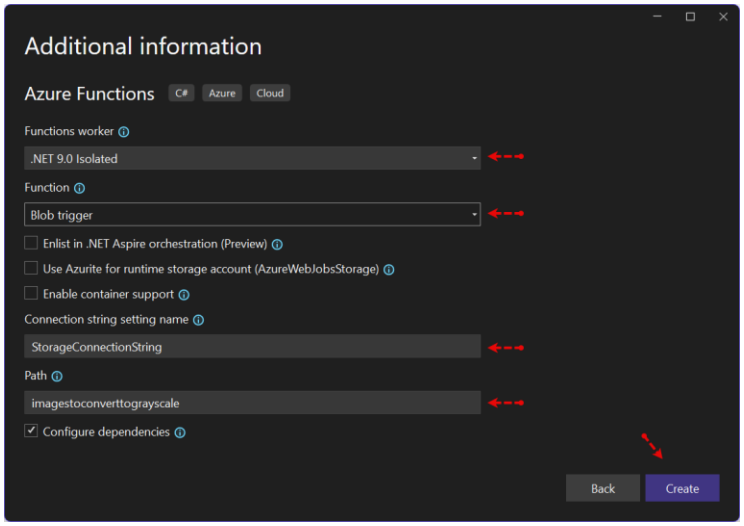
- Caveats
  - There is no built-in poison “blob” handling
  - Consumption plan
    - May have up to a 10-minute delay in notifying you
  - Blobs don’t automatically get removed
  - Need to be creative in handling “meta-data”
    - Use separate containers
      - To process blobs differently
    - Use blob metadata
      - To provide additional processing information

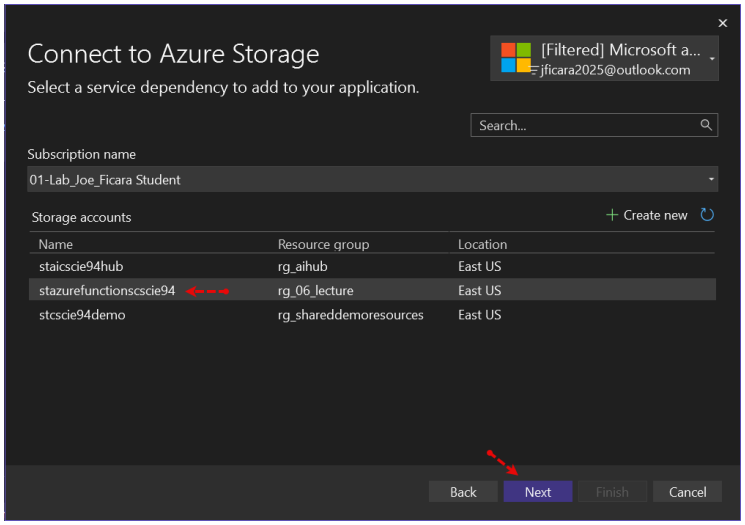
Blob Trigger

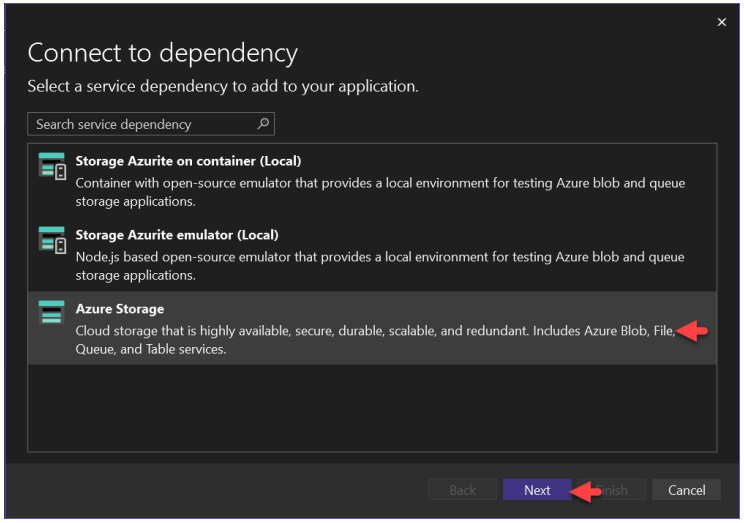
- When creating the azure function project
  - Choose
    - Blob trigger
    - Storage account
    - Name for storage account connection string
    - Path to monitor (Container…)
  - Notes:
    - Can also add functions to an existing project

Blob Trigger

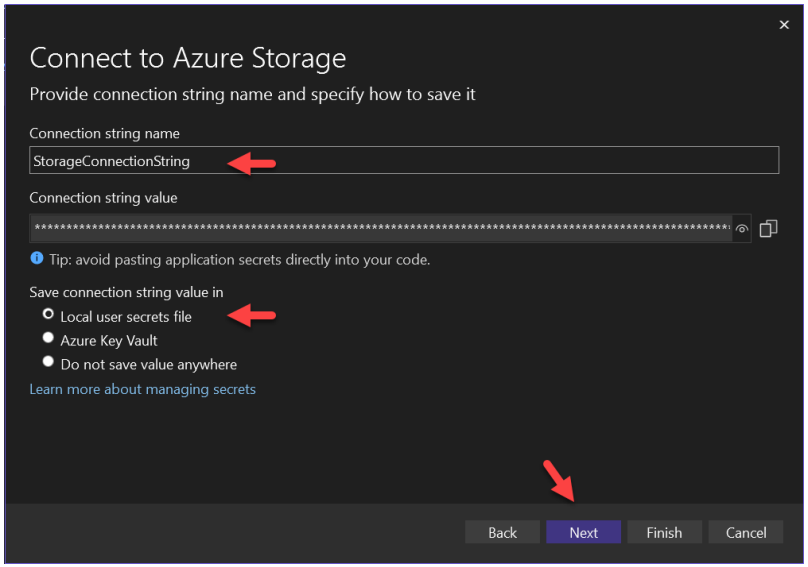


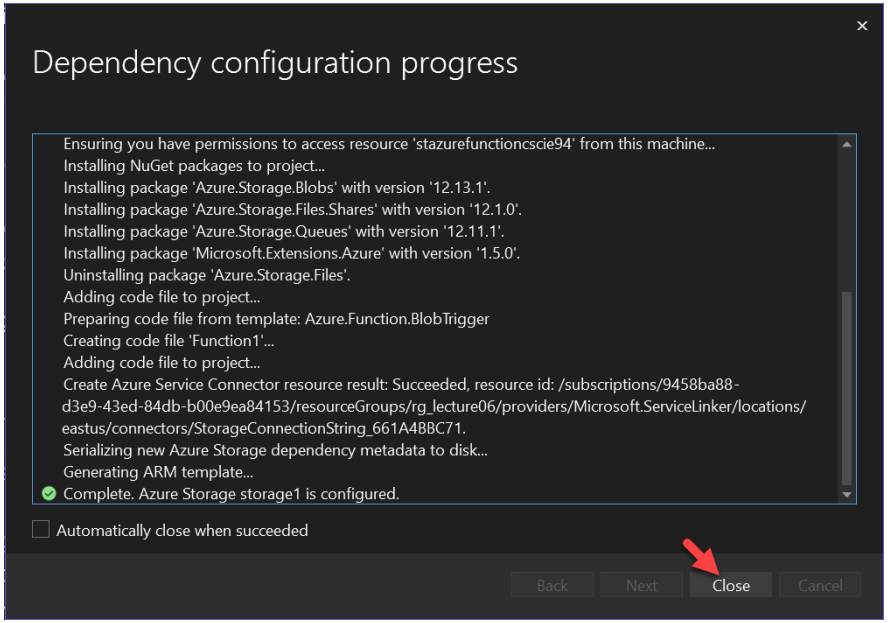






Blob Trigger





Demo

Blob Triggers

`BlobTriggerDemoSolution`

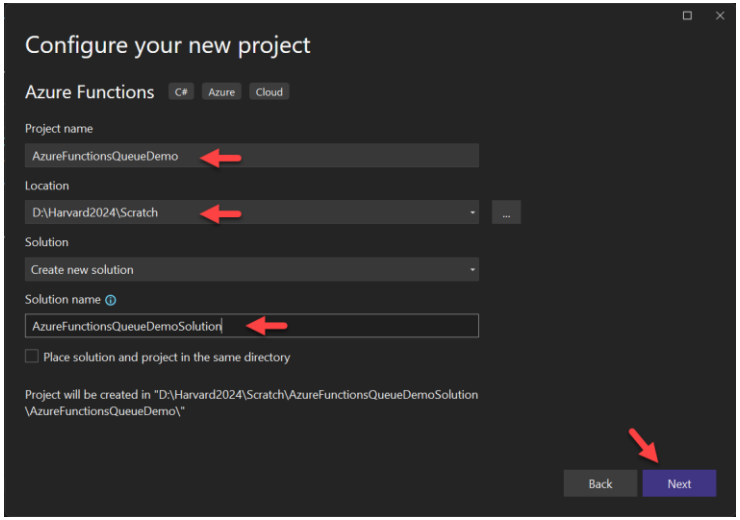
Queue Trigger

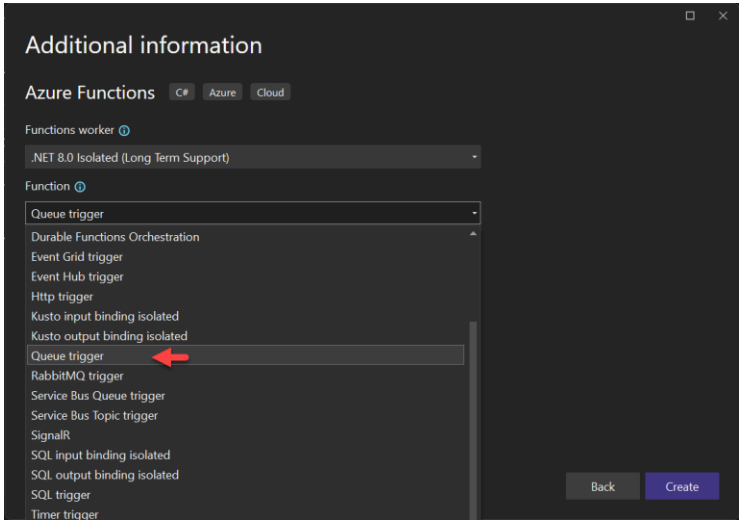
- Same mental model as web jobs
  - Message arrives in a Queue
  - Function is invoked
    - Process the message
- Automatic poison message handling
  - A queue is automatically created
    - <*your queue name*>-poison
    - Poison message is put in that queue

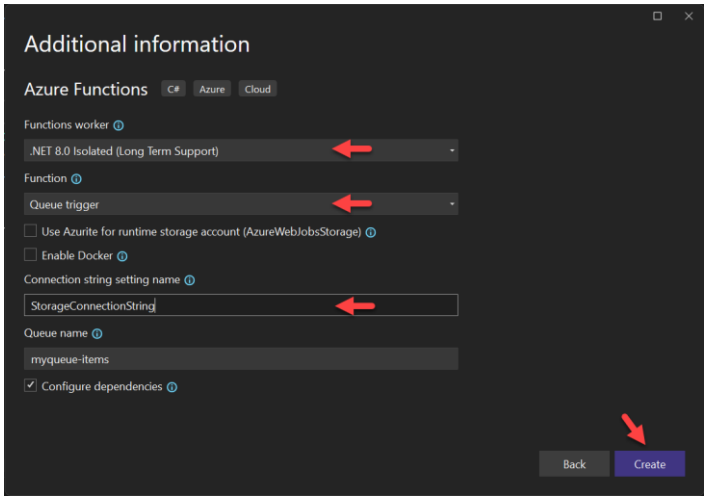
Queue Trigger

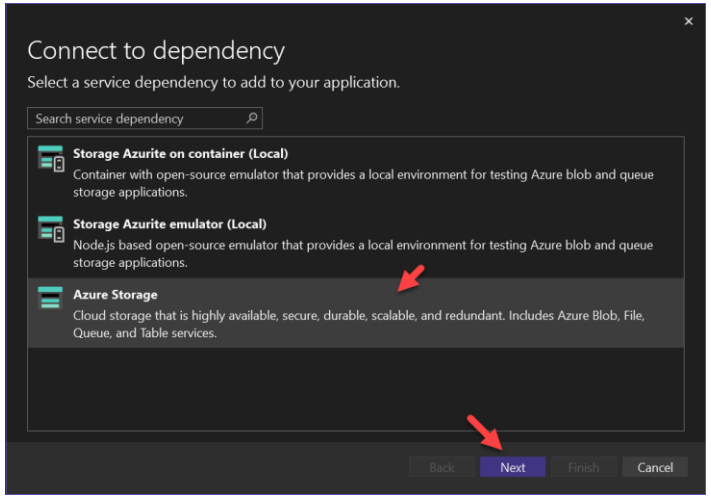
- By default, the poison message count is 5
  - Meaning if the message fails 5 times
    - It moves into the poison queue
- Configuration is done in the host.json file
  host.json reference for Azure Functions 2.x and later
    - maxPollingInterval
    - visibiltiyTimeout
    - batchSize
    - maxDequeueCount
    - newBatchThreshold

Queue Trigger

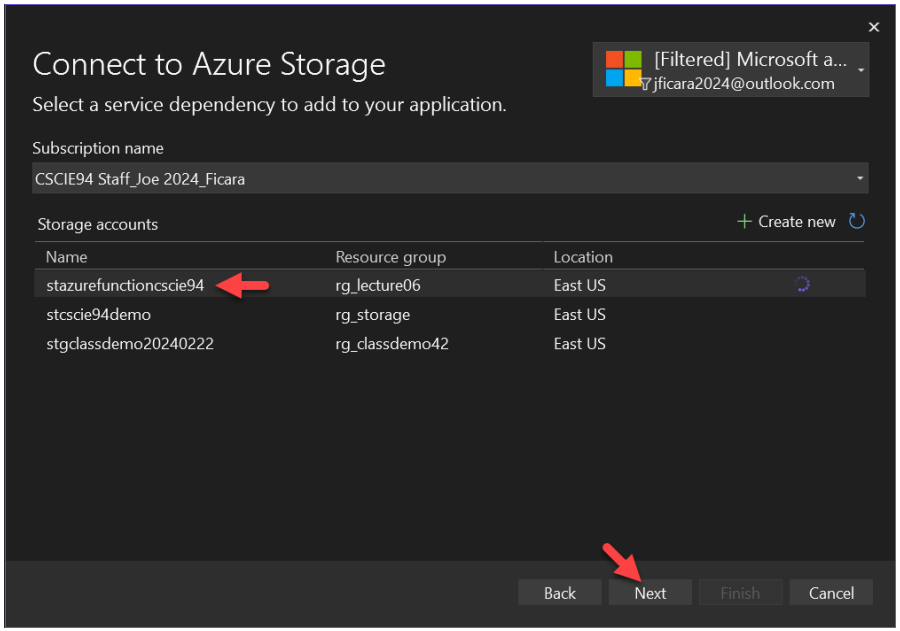


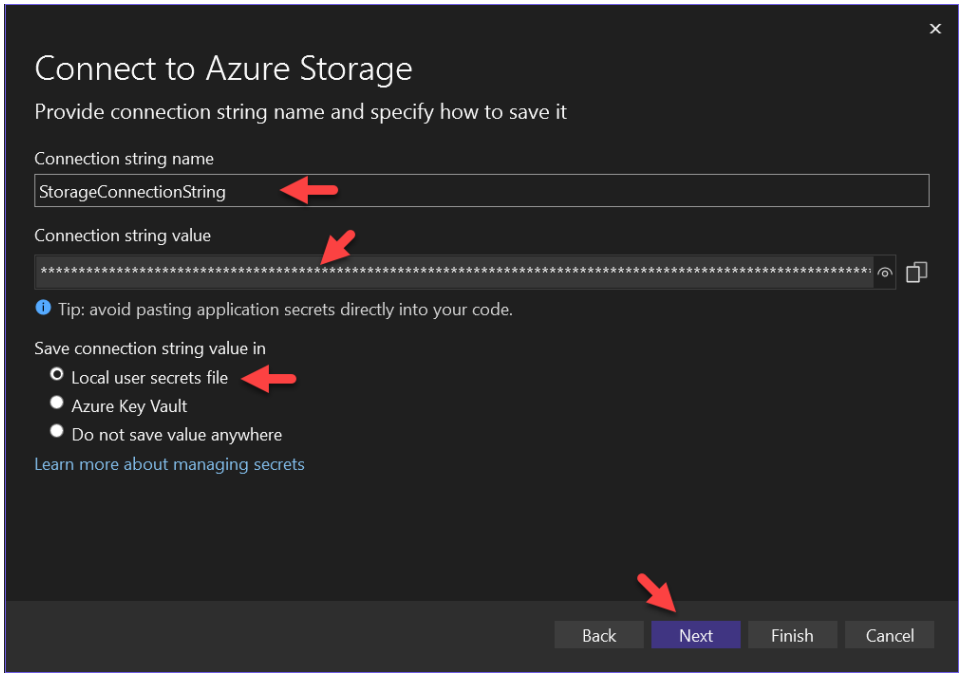




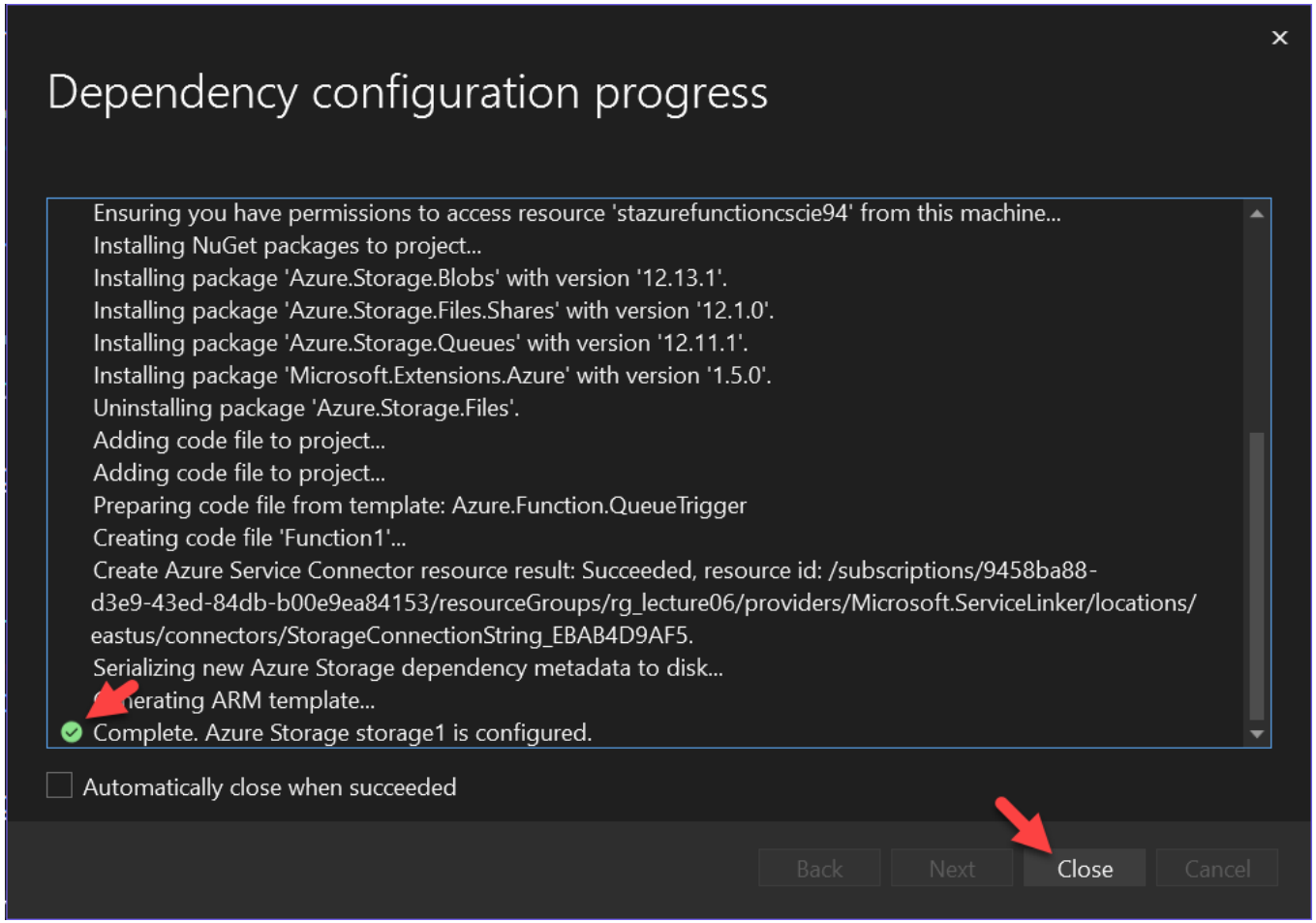


Queue Trigger





Queue Trigger



Demo

Queue Trigger

`QueueTriggerDemoSolution`

Timer Trigger

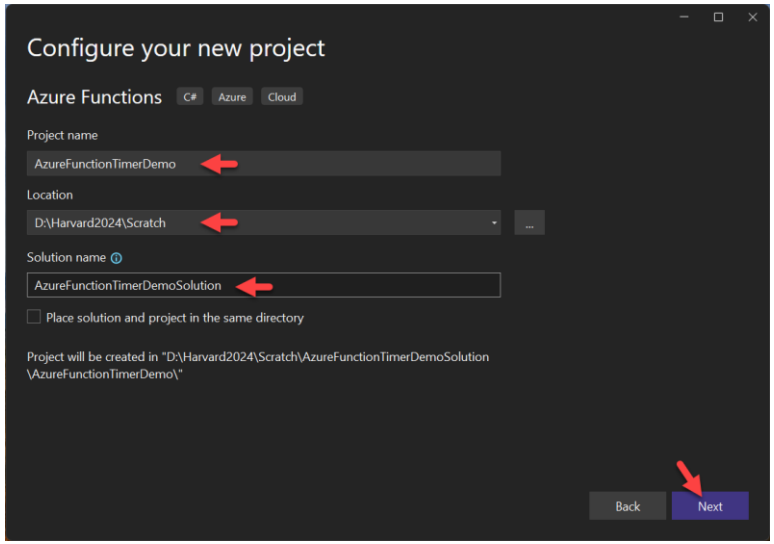
- Mental Model
  - Think scheduled web jobs
- The Azure Scheduler is not used
  - But you could if you wanted to
    - How might you do that?
- How does it work?
  - Use the TimerTrigger

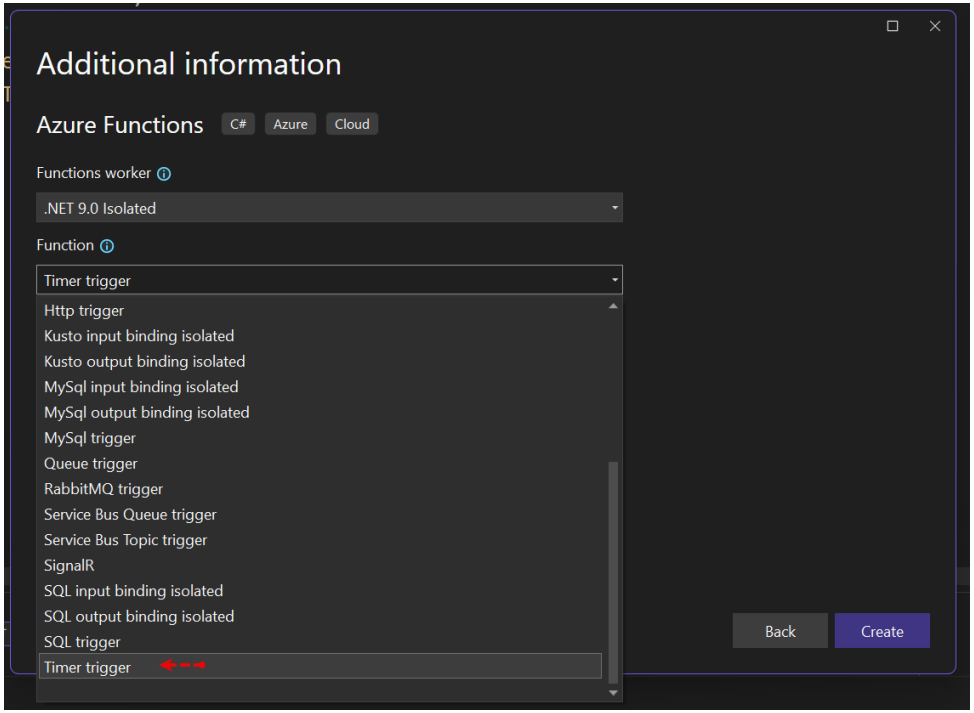
Timer Trigger

- You specify
  - A CRON expression
    - Indicates the schedule the function will be invoked
  - Format:
    - {second} {minute} {hour} {day} {month} {day-of-week}
  - Example:
    - */5 * * * * *
      - Indicates run every 5 seconds
      - https://en.wikipedia.org/wiki/Cron#CRON_expression

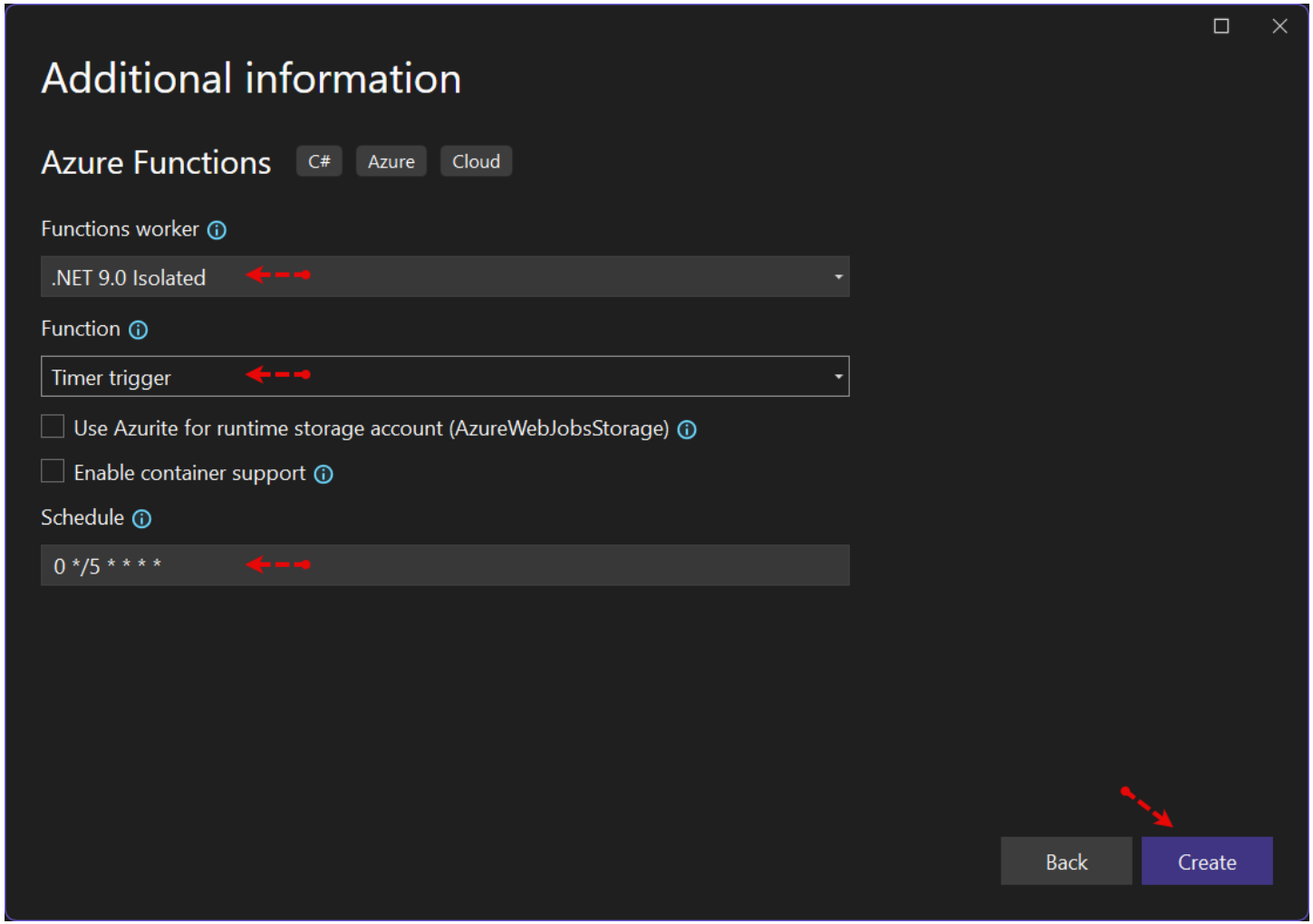
Timer Trigger

- To Create a Timer Trigger Azure Function





Timer Trigger



Demo

Timer Triggers

`AzureFunctionTimerDemoSolution`

Questions ?

Further Reading

- Creating your first function app
  - Getting started with Azure Functions
- Azure Functions Documentation
  - Azure Functions documentation
- Best practices for scaling
  - Best practices for reliable Azure Functions
- Function Scaling
  - Azure Functions hosting options

Further Reading

- HTTP Client usage notes
  - Improper Instantiation antipattern
- host.json reference for azure functions
  - host.json reference for Azure Functions 2.x and later
- Azure Functions F# Developer Reference
  - Using Azure Functions with F#
    - Note: The F# documentation seems to be missing as of 2025-03-05

Further Reading

- Azure Functions runtime version overview
  - Azure Functions runtime versions overview