Microsoft

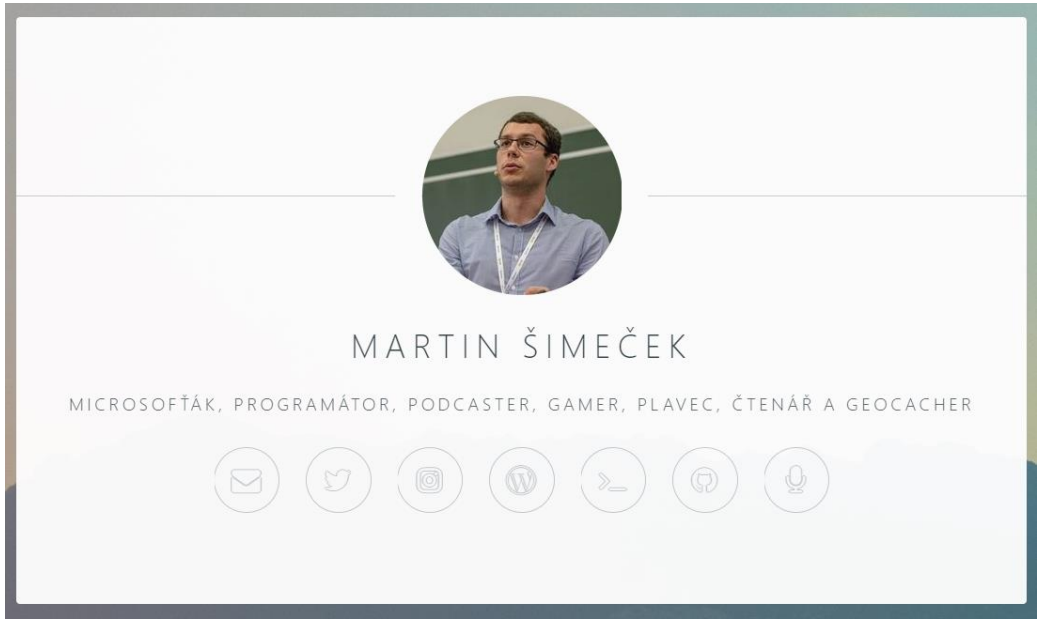
Azure Functions prakticky

Martin Šimeček
@deeedx
martin.simecek@microsoft.com

whoami?

5 years @ Microsoft

Consultant > Technical Evangelist > Software Engineer



http://deedx.cz

.NET.CZ Podcast

http://bit.ly/netczpodcast

Azure Functions

Azure Functions

Serverless

Event-driven scale

Reduced Dev Ops

Accelerate development

Develop your way

Local development

Bind into services

Azure Service Bus

Azure Event Hub

Azure Storage

Dropbox

Sendgrid

Azure CosmosDB

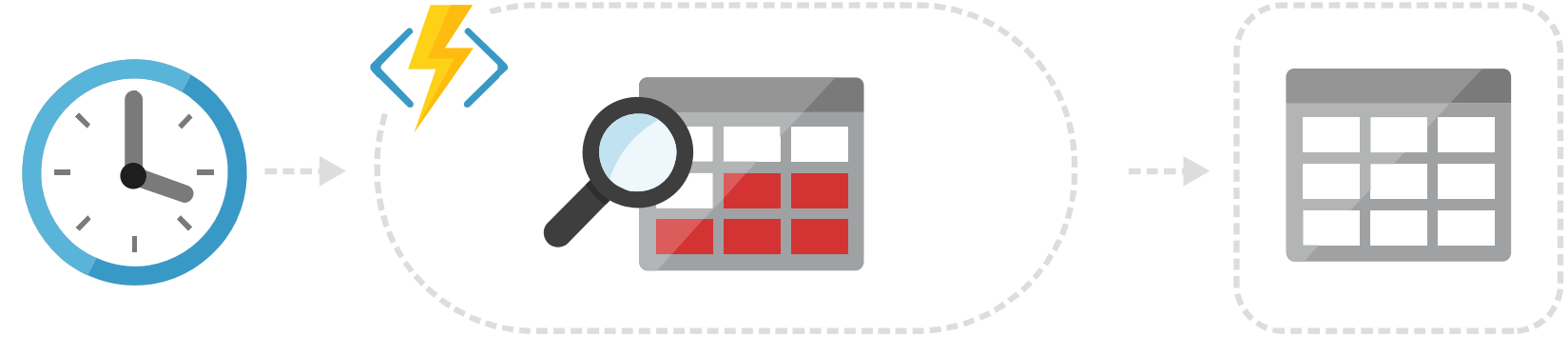
OneDrive

Box

Twilio

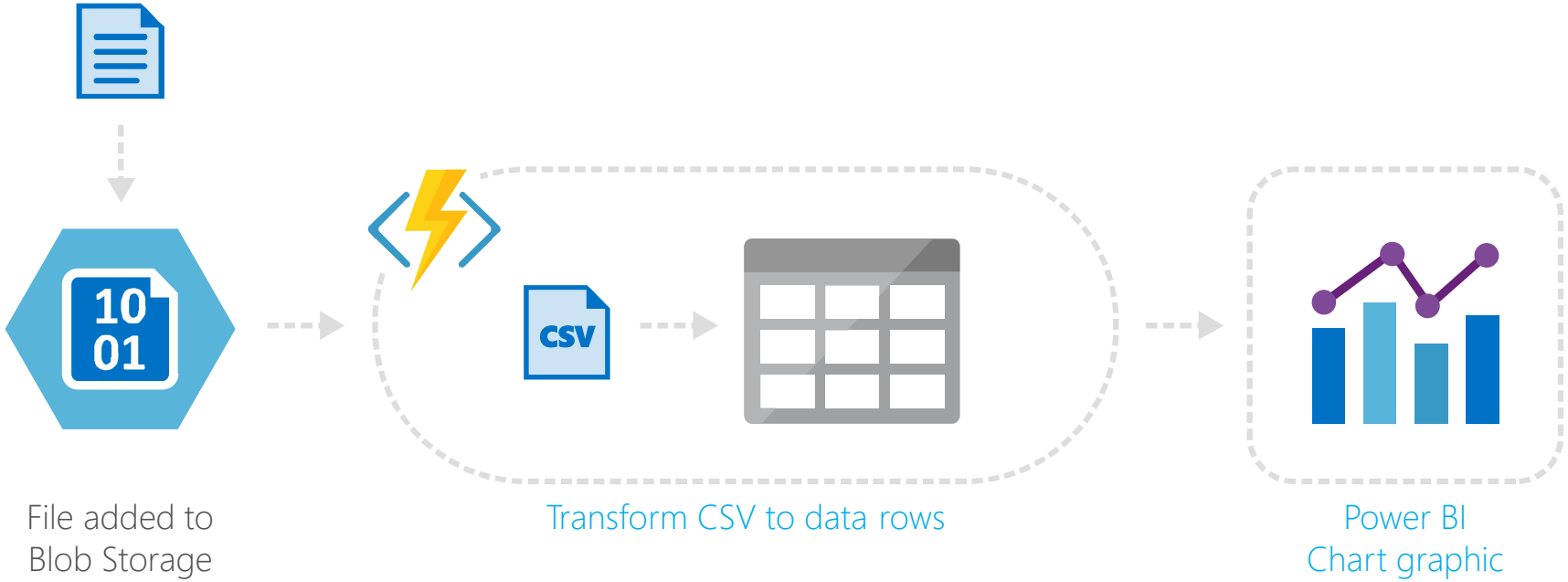
Example

Timer based processing



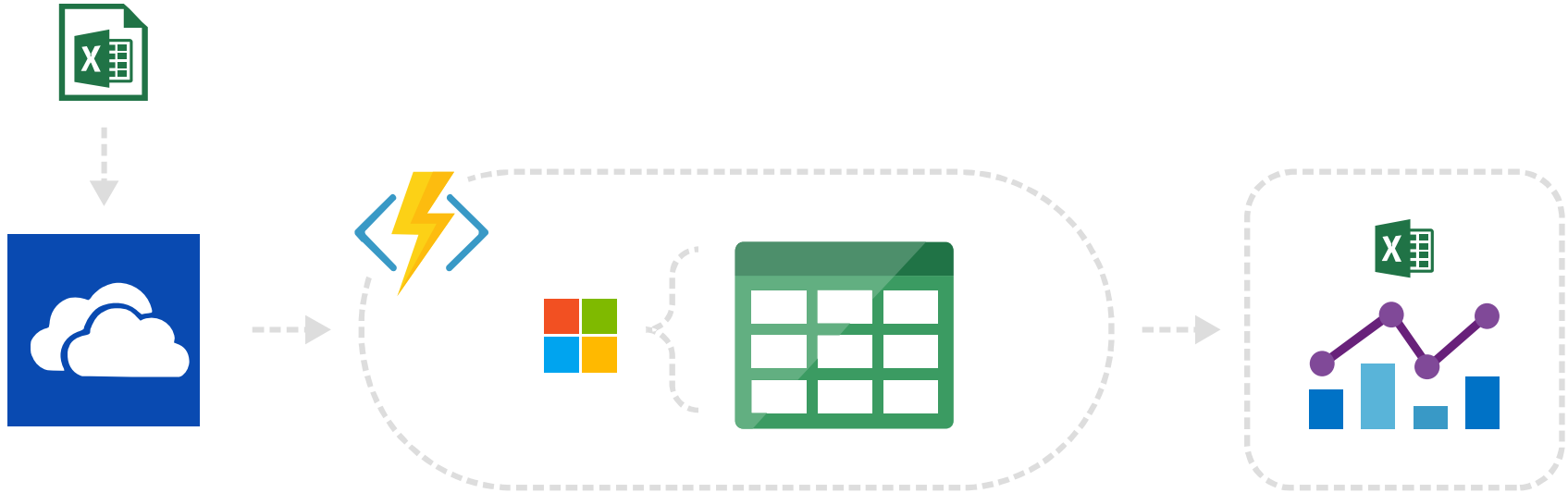
Example

Azure service event processing



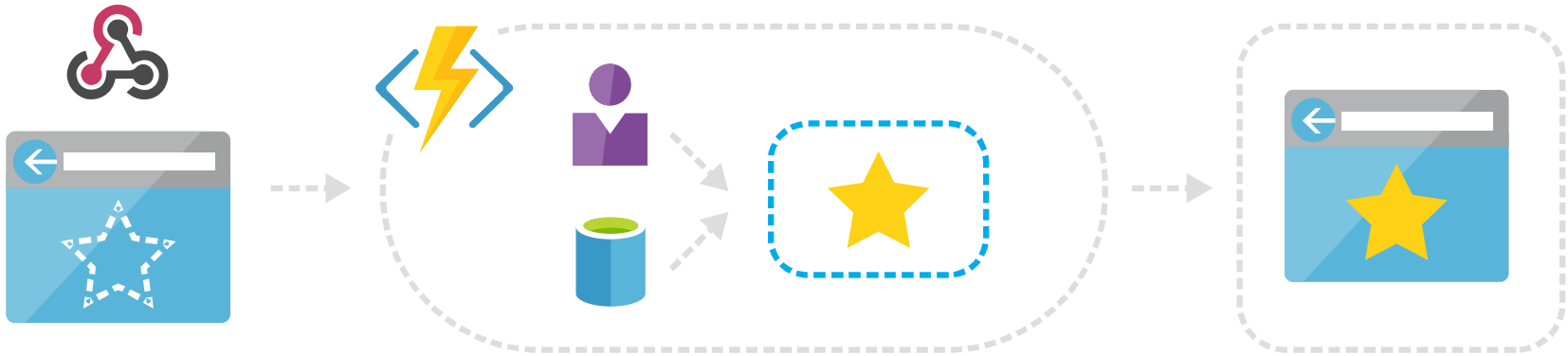
Example

SaaS event processing



Example

Serverless Web Applications architectures



Example

Real-time stream processing

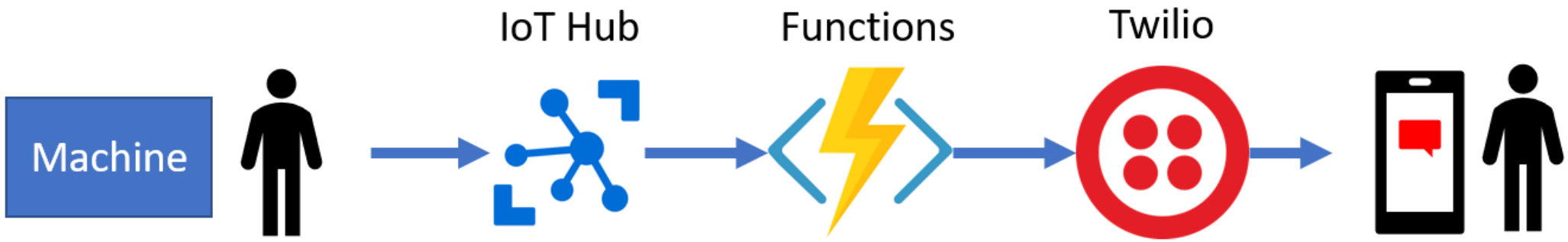
Millions of devices feed into Stream Analytics

Transform to structured data

Store data in SQL Online

Case

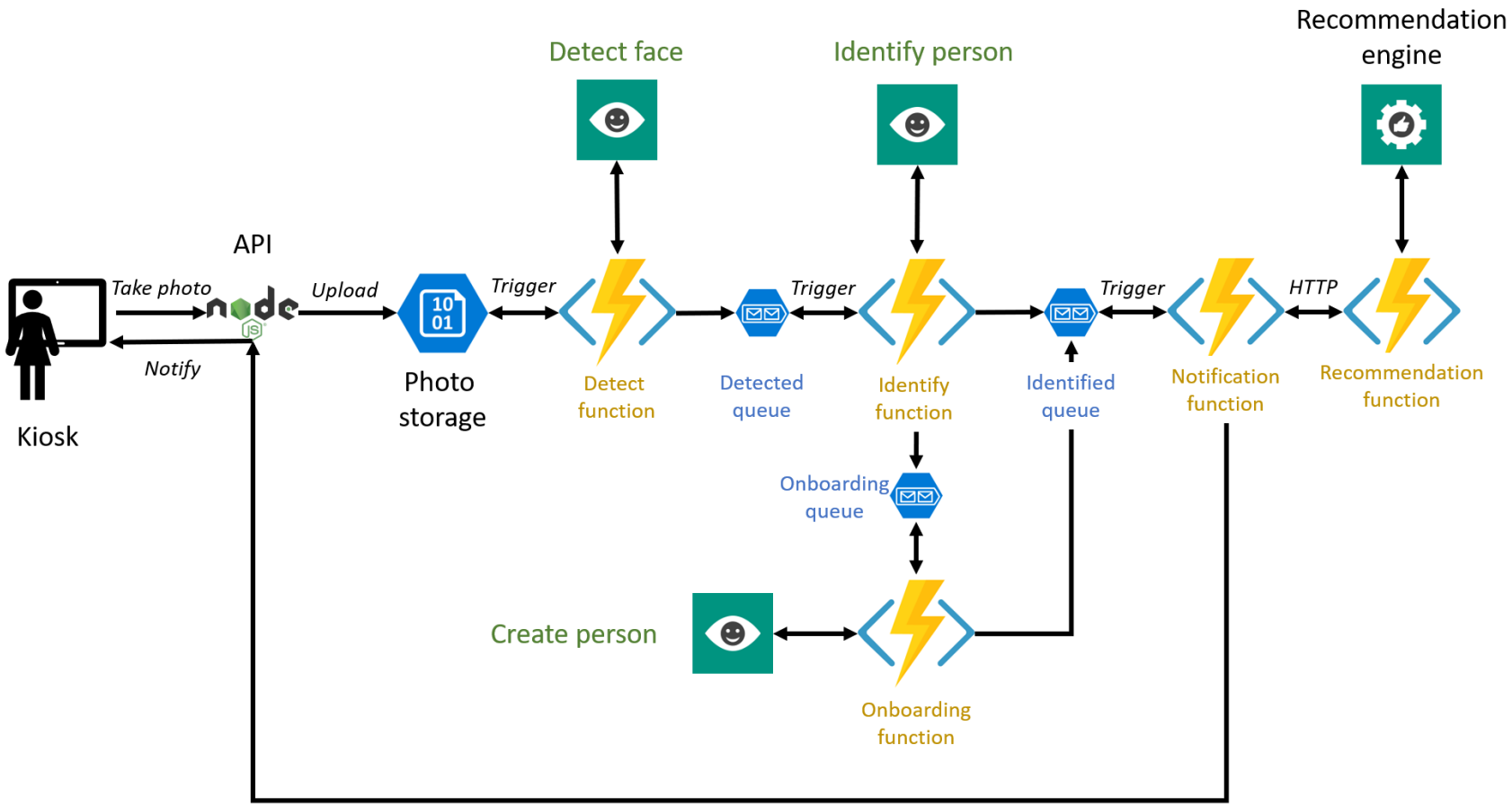
SMS initiated by IoT Hub



https://codez.deedx.cz/posts/from-iot-hub-to-sms/

Case

Self-service kiosk



Tips, tricks & learnings

V1 vs. V2

V1

- GA
- .NET Framework
- C#, JavaScript, F#

V2

- Preview
- .NET Core
- C#, JavaScript, Java
- language extensibility

open-source

Hosting

App Service Plan
Consumption Plan

??

Hosting

App Service Plan

Hosted as Azure App Service
Paid 24/7
Always On
Predictable performance & cost
=> Efficient for steady workloads

Consumption Plan

Hosting

App Service Plan

Consumption Plan

Special type of App Service Plan
Paid per execution & consumed memory
Cold starting
Complex cost estimates
=> Efficient for spike workloads

Bindings

Basics

Type	1.x	2.x	Trigger	Input	Output
Blob Storage	✔	✔	✔	✔	✔
Cosmos DB	✔	✔	✔	✔	✔
Event Grid	✔	✔	✔		
Event Hubs	✔	✔	✔		✔
HTTP	✔	✔	✔		✔
Microsoft Graph Excel tables		✔		✔	✔
Microsoft Graph OneDrive files		✔		✔	✔
Microsoft Graph Outlook email		✔			✔
Microsoft Graph Events		✔	✔	✔	✔
Microsoft Graph Auth tokens		✔		✔	
Mobile Apps	✔	✔		✔	✔
Notification Hubs	✔				✔
Queue storage	✔	✔	✔		✔
SendGrid	✔	✔			✔
Service Bus	✔	✔	✔		✔
Table storage	✔	✔		✔	✔
Timer	✔	✔	✔		
Twilio	✔	✔			✔
Webhooks	✔		✔		✔

https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings

Syntax

```
[QueueTrigger("detected-faces", Connection = "ImageStorage")]DetectMessage detectedFaceMessage,
[Table("inputTable", "{name}", "{rand-guid}", Connection = "ImagesConnection")] MappingEntity inputTable,
[Queue("identified-faces", Connection = "ImageStorage")] out IdentifyMessage identifiedOutput,
[Queue("onboarding-faces", Connection = "ImageStorage")] out OnboardingMessage onboardingOutput,
```

1. *When a message is enqueued, parse it and load entity from Storage Table.*
2. *Do work.*
3. *Enqueue message to one (or both) of the output queues.*

The Power of Bindings #1

```
[FunctionName("BlobTrigger")]
public static void Run(
    [BlobTrigger("images/{name}", Connection = "ImagesConnection")]Stream myBlob,
    string name,
    TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {myBlob.Length} Bytes");
}
```

download blob automatically

assign dynamic value to parameter

The Power of Bindings #1

```
[FunctionName("BlobTrigger")]
public static void Run(
    [BlobTrigger("images/{name}", Connection = "ImagesConnection")]Stream myBlob,
    string name,
    TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {myBlob.Length} Bytes");
}
```

```
[QueueTrigger("filenames", Connection= "ImagesConnection")] FileMessage queueMessage,
[Blob("images/{Filename}", FileAccess.Write, Connection= "ImagesConnection")] Stream outputImage,
```

parse strongly-typed

output

property of trigger object

```
public class FileMessage
{
    public string Url { get; set; }
    public string Filename { get; set; }
}
```

The Power of Bindings #1

```
[FunctionName("BlobTrigger")]
public static void Run(
    [BlobTrigger("images/{name}", Connection = "ImagesConnection")]Stream myBlob,
    string name,
    TraceWriter log)
{
    log.Info($"C# Blob trigger function Processed blob\n Name:{name} \n Size: {myBlob.Length} Bytes");
}
```

```
[QueueTrigger("filenames", Connection= "ImagesConnection")] FileMessage queueMessage,
[Blob("images/{Filename}", FileAccess.Write, Connection= "ImagesConnection")] Stream outputImage,
```

```
public class FileMessage
{
    public string Url { get; set; }
    public string Filename { get; set; }
}
```

---

custom HTTP route

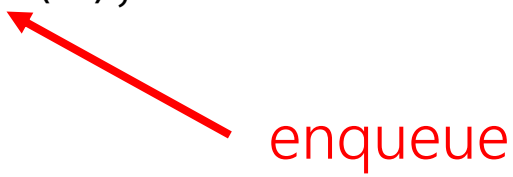
```
[HttpTrigger(AuthorizationLevel.Anonymous, "get", Route = "track/{tableName}")]HttpRequestMessage req,
[Table("{tableName}", Connection = "AzureWebJobsStorage")] out TableRow tableOutput)
```

The Power of Bindings #2

ICollector

```
[Queue("filenames", Connection="MyQueueConnection")] ICollector<string> queue

foreach (string f in files)
{
     queue.Add(f);
}
```



The Power of Bindings #2

Single message

```
[EventHubTrigger("hub")] string eventHubMessage
[EventHubTrigger("hub")] EventData eventHubMessage
```

Batch

```
[EventHubTrigger("hub")] string[] eventHubMessage
[EventHubTrigger("hub")] EventData[] eventHubMessage
```

host.json:

```
{
 "eventHub":
 {
  "maxBatchSize": 64,
  "prefetchCount": 256,
  "batchCheckpointFrequency": 1
 }
}
```

The Power of Bindings #3

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] Stream image
```

vs.

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] CloudBlockBlob image
```

The Power of Bindings #3

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] Stream image
```

VS.

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] CloudBlockBlob image
```

```
image.CopyTo(outputImage);
outputImage.Dispose();
```

commit changes

The Power of Bindings #3

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] Stream image
```

VS.

```
[BlobTrigger("photos/{name}", Connection = "ImageStorage")] CloudBlockBlob image
```

```
image.Metadata.TryGetValue("kioskId", out string kioskId)
```

read blob metadata

```
var imageStream = new MemoryStream();
await image.DownloadToStreamAsync(imageStream);
imageStream.Seek(0, SeekOrigin.Begin);
```

download blob manually

The Power of Bindings #4

IQueryable

no PartitionKey & RowKey specified

```
[Table("inputTable", Connection = "ImagesConnection")] IQueryable<MappingEntity> inputTable
```

```
List<MappingEntity> res = inputTable.Where(p => p.PartitionKey == "PK" && p.RowKey == name).ToList();
```

using LINQ to get entity in code

The Power of Bindings #5

Binding Extensions in v2

Extensibility model for the SDK

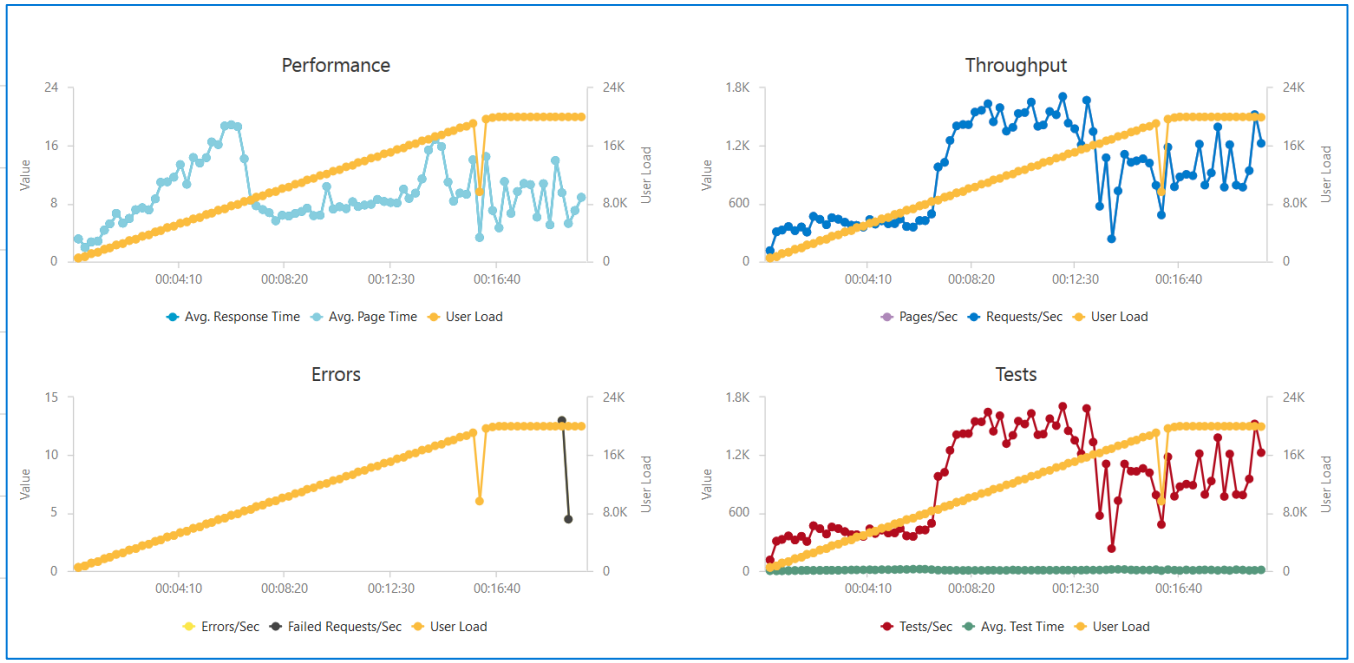
Build your own Triggers, Inputs & Outputs

Performance

Superscale

1000 HTTP RPS, Consumption Plan

Total requests	1 097 659
Failed requests	262
Max servers used	35
Average RPS	**914**
Max RPS	**1711**
Average response time	8.7 s
Estimated price (20 minutes)	$5.75
Estimated price (1 hour)	$17.25

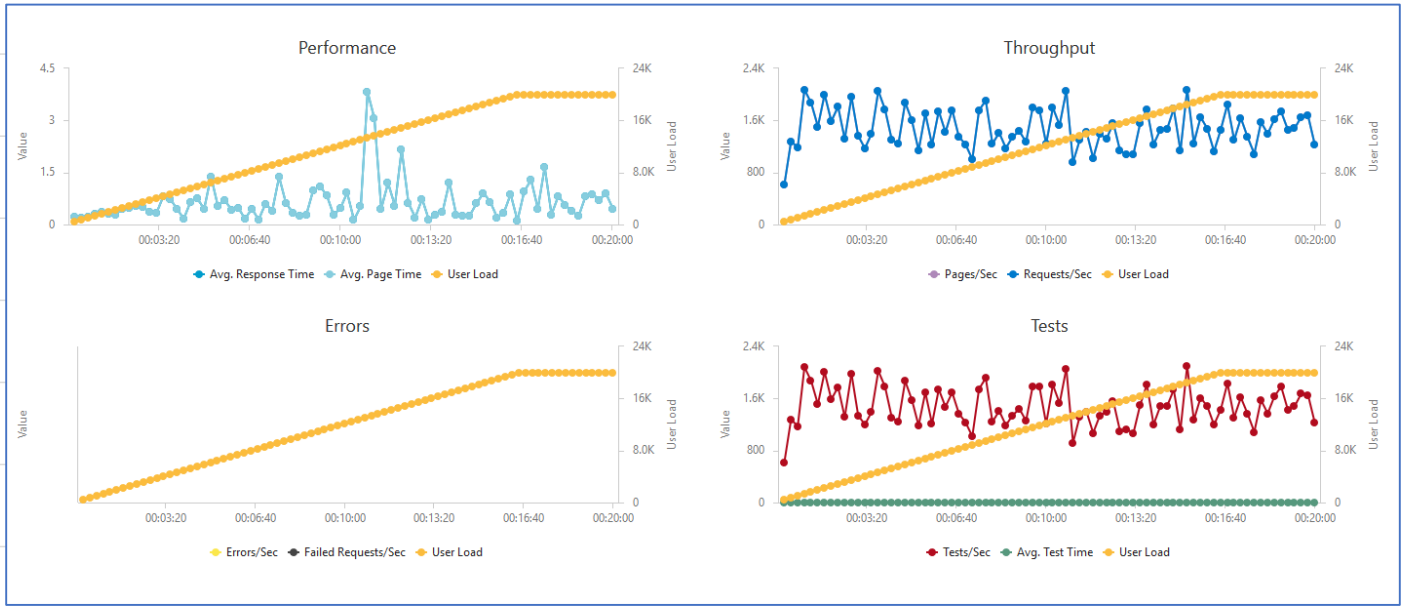


https://codez.deedx.cz/projects/high-throughput-azure-functions-on-http/

Superscale

1000 HTTP RPS, P3 Plan, 20 instances

Total requests	1 770 287
Failed requests	**0**
Max servers used	20
Average RPS	1475
Max RPS	**2075**
Average response time	**0.63 s**
Estimated price (20 minutes)	$7.6
Estimated price (1 hour)	$23



https://codez.deedx.cz/projects/high-throughput-azure-functions-on-http/

Logging

Disable TraceWriter -> use Application Insights

Remove `AzureWebJobsDashboard` app setting

Add `APPINSIGHTS_INSTRUMENTATIONKEY`

Megascale

100 000 RPS

Use Event Hub & request batching

=> reduce Function execution overhead

```
[EventHubTrigger("hub", Connection = "EventsConnection")] string[] eventHubMessage
[EventHubTrigger("hub", Connection = "EventsConnection")] EventData[] eventHubMessage
```

https://blogs.msdn.microsoft.com/appserviceteam/2017/09/19/processing-100000-events-per-second-on-azure-functions/

Storage

Use different Storage Account for high-load scenarios

AzureWebJobsStorage
ImageStorage

Avoid Storage completely for „mega“ scale

Runtime

Run-From-Zip (Preview)

Node.js cold-start can take long…

https://itnext.io/building-single-page-app-with-azure-functions-and-improving-cold-start-time-79a0faec9913

Zip the app and let Azure mount it as a read-only file system:

```
WEBSITE_USE_ZIP=https://mujstorage.blob.core.windows.net/site/MyApp.zip?st=2018-02-
<...SAS token>
```

Applies to Web Apps too ;)

https://github.com/Azure/app-service-announcements/issues/84

Funcpack

Uses **webpack** to bundle all Node modules into a single file

```
npm install -g azure-functions-pack
funcpack pack ./
```

functions.json is then using this bundle

hosts.json

Event Hub batch size

HTTP route prefix

HTTP max outstanding requests

HTTP max concurrent requests

Logger level

Queue polling interval

https://docs.microsoft.com/en-us/azure/azure-functions/functions-host-json

Recommendations

Do

Build small functions

Avoid long-running functions

Respond fast

Use queues for cross-function communications

Write functions to be stateless

Use Docs (https://docs.microsoft.com/en-us/azure/azure-functions/)

Microsoft

Azure Functions prakticky

Martin Šimeček
@deeedx
martin.simecek@microsoft.com