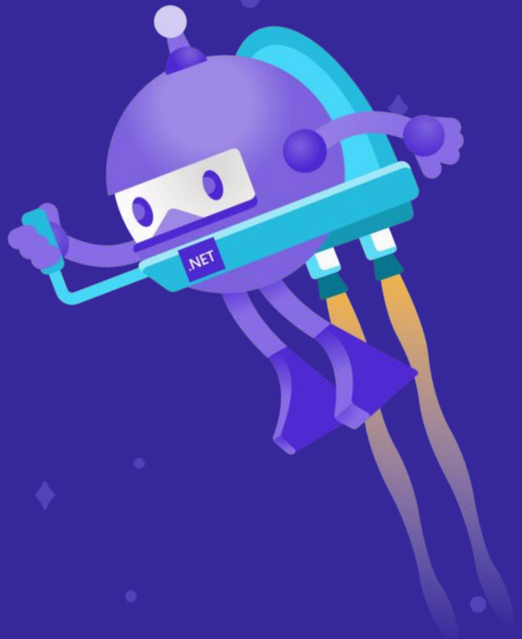
# Welcome to .NET Conf!
# by DotNetToscana

# The art of Azure Functions (unit) testing and monitoring

Massimo Bonanni

Azure Technical Trainer

@massimobonanni



## The issue....

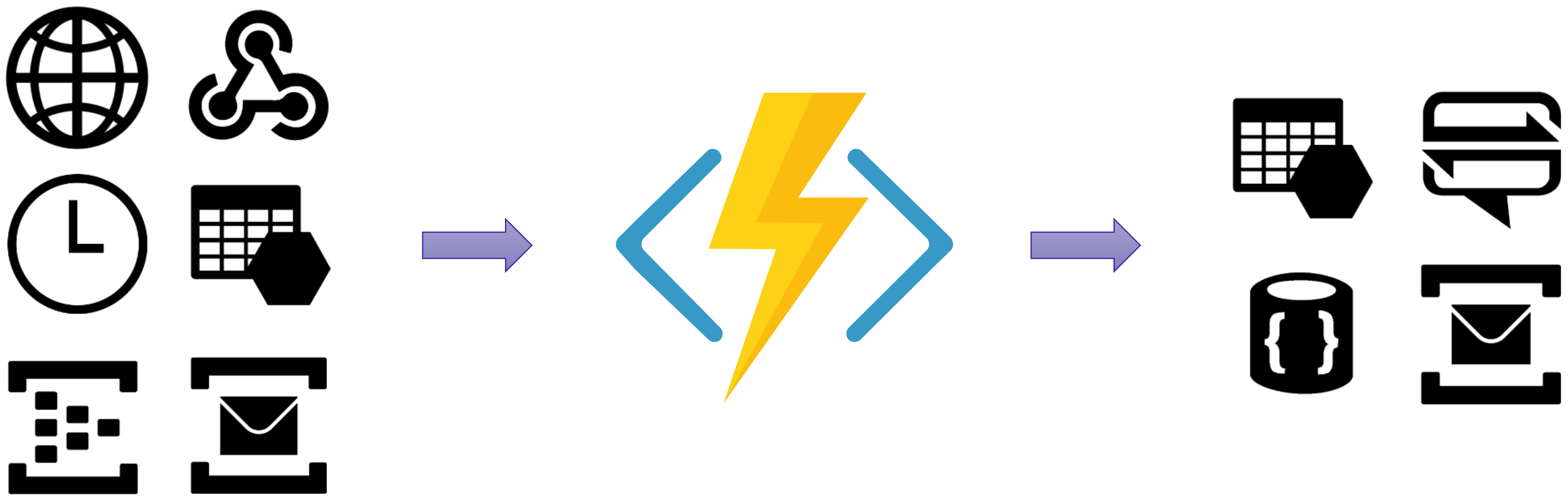
If you want to use **Azure Functions** as components of your **Enterprise solutions**, you **must** to test and monitor them!!!

# What are Azure Functions

## Events

## Code

## Outputs



React to timers, HTTP, or events from your favorite Azure services, with more on the way

Author functions in C#, F#, Node.JS, Java, Powershell, and more

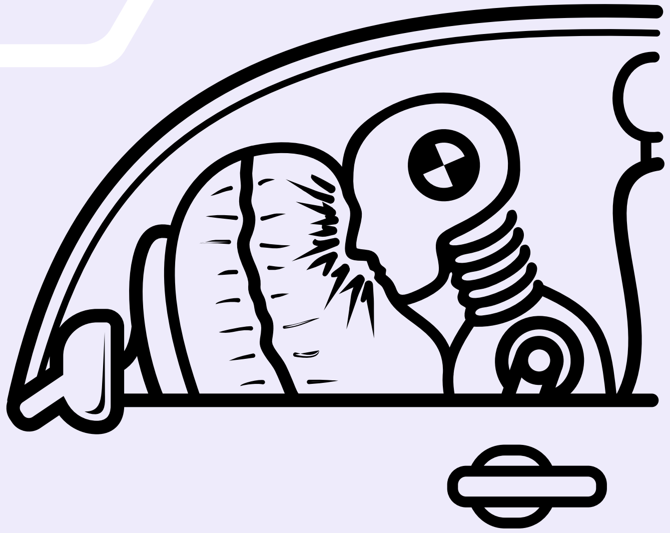
Send results to an ever-growing collection of services

## What is a Unit Test

In a **unit test** you invoke a piece of your code with a set of parameters and you checks the correctness its behavior.

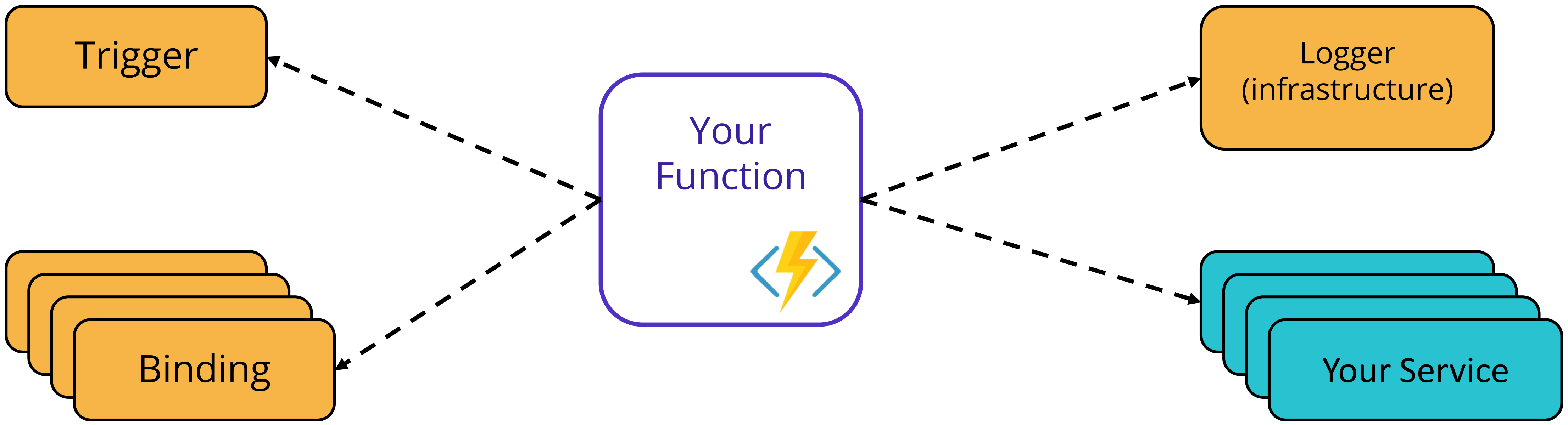
In a **unit test** you must substitute all your external reference with a **mock** or **stub**.

***Mock*** *is for the software what a* ***dummy*** *is for a car crash test (you don’t test a car with a human being inside…I Hope!!)*

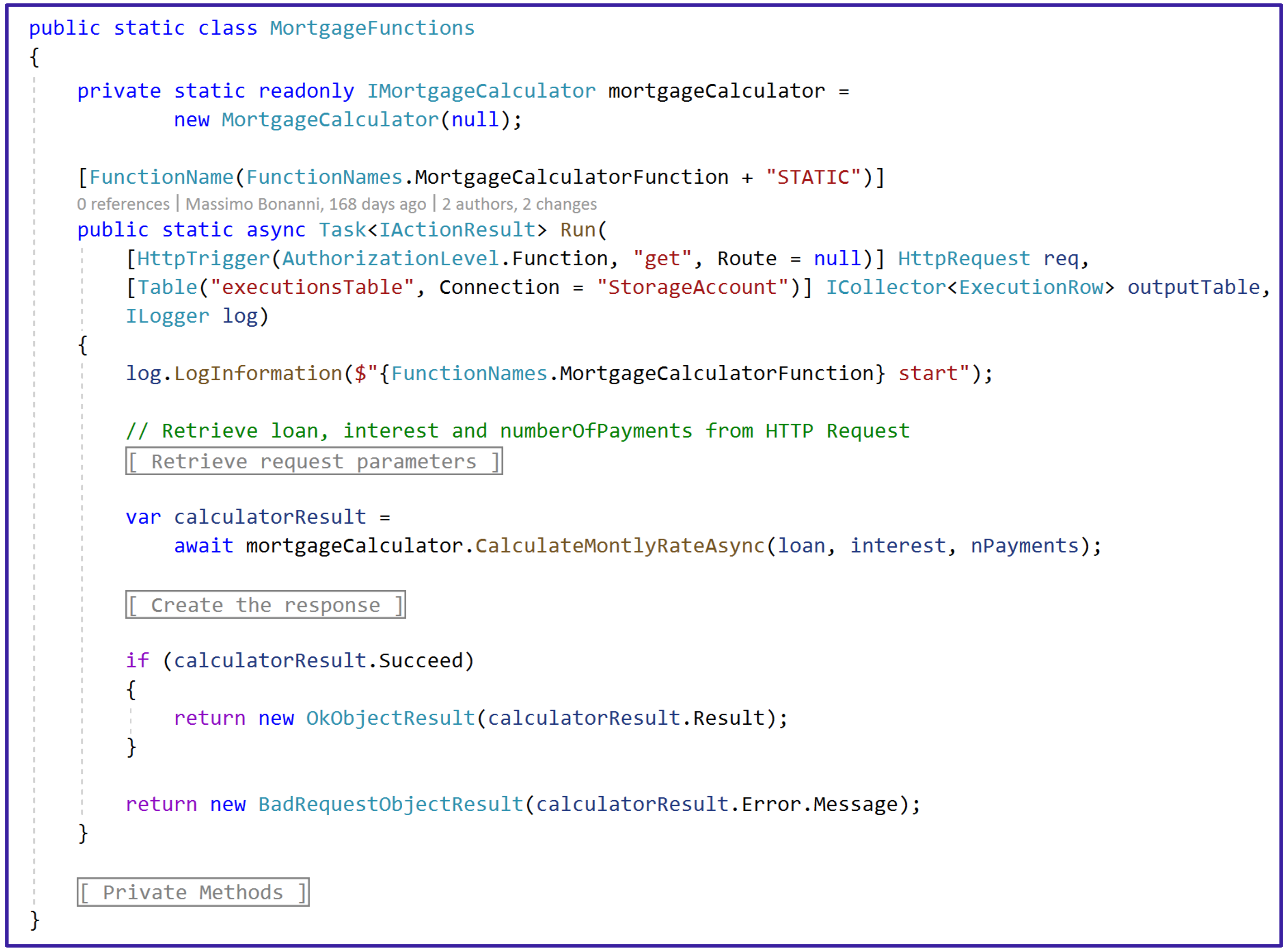


# Azure Functions Dependencies

You **should implement** your Azure Functions to allow you to use mock/stub for all external reference!

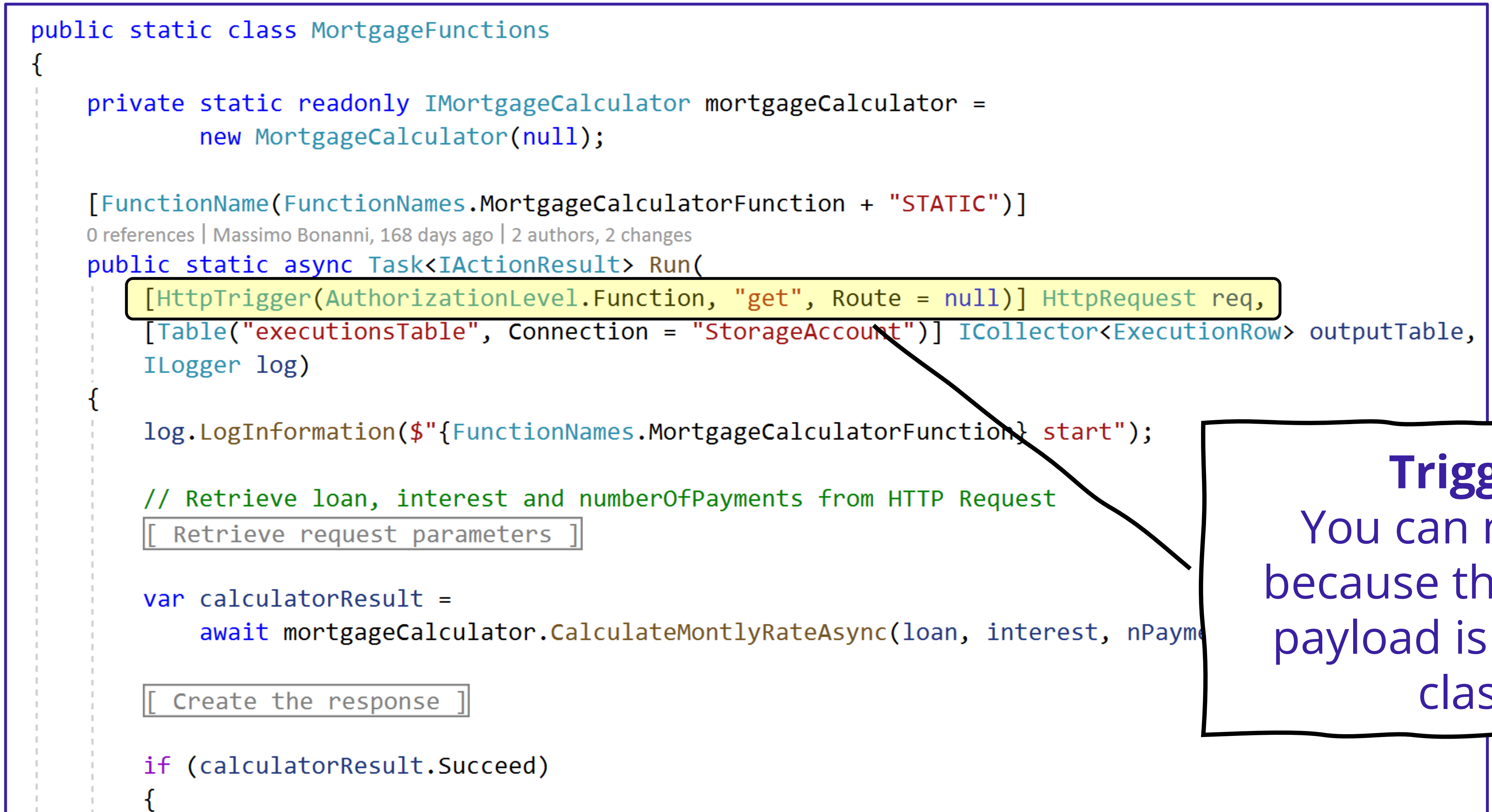


# Azure Function … untestable!!



```
public static class MortgageFunctions
{
    private static readonly IMortgageCalculator mortgageCalculator =
            new MortgageCalculator(null);

    [FunctionName(FunctionNames.MortgageCalculatorFunction + "STATIC")]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 2 changes
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} start");

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await mortgageCalculator.CalculateMontlyRateAsync(loan, interest, nPayments);

        [ Create the response ]

        if (calculatorResult.Succeed)
        {
            return new OkObjectResult(calculatorResult.Result);
        }

        return new BadRequestObjectResult(calculatorResult.Error.Message);
    }

    [ Private Methods ]
}
```

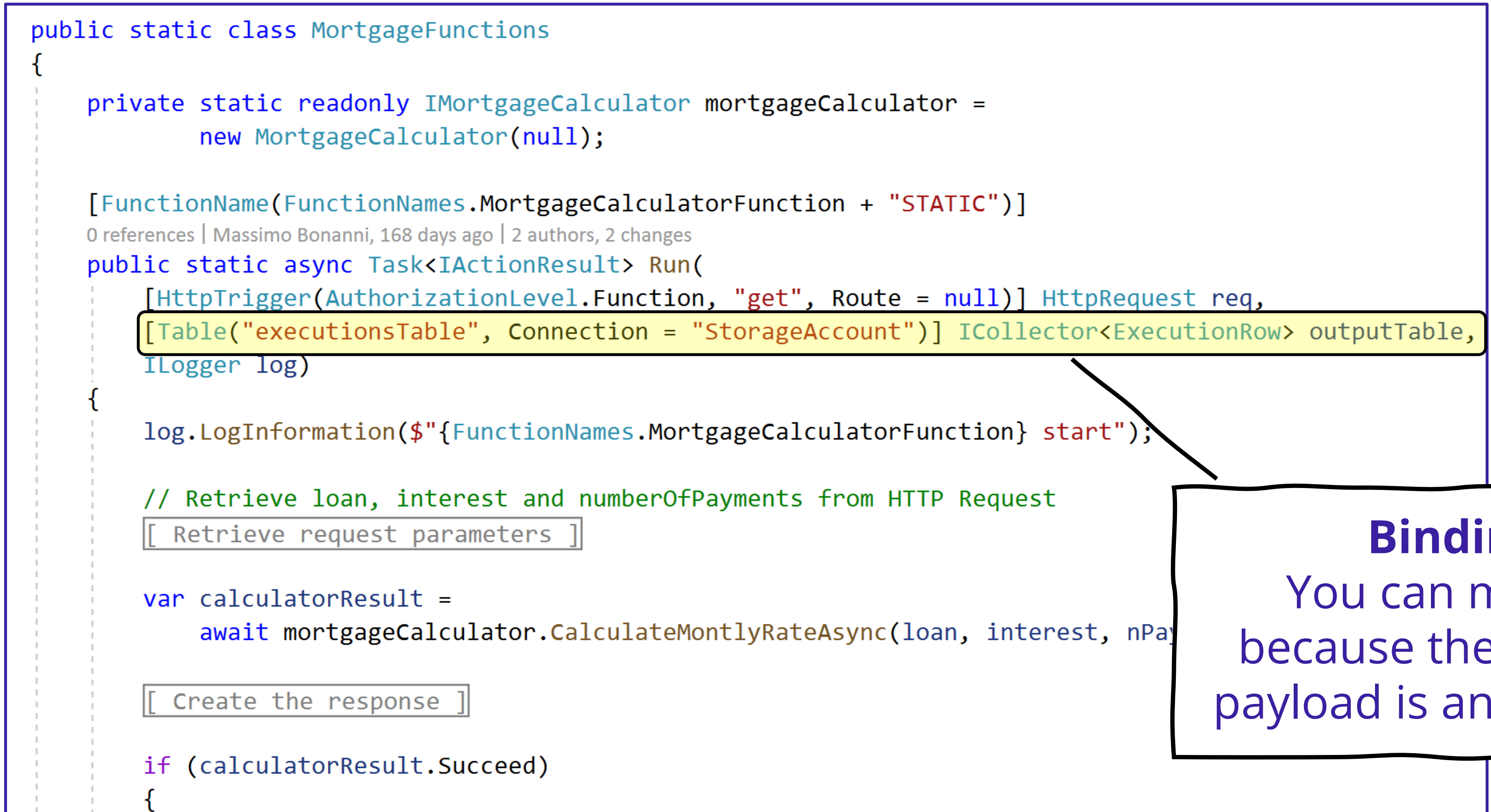
# Azure Function … trigger!!



```
public static class MortgageFunctions
{
    private static readonly IMortgageCalculator mortgageCalculator =
            new MortgageCalculator(null);

    [FunctionName(FunctionNames.MortgageCalculatorFunction + "STATIC")]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 2 changes
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} start");

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await mortgageCalculator.CalculateMontlyRateAsync(loan, interest, nPayme

        [ Create the response ]

        if (calculatorResult.Succeed)
        {
```

**Trigger**
You can mock it because the trigger payload is a POCO class

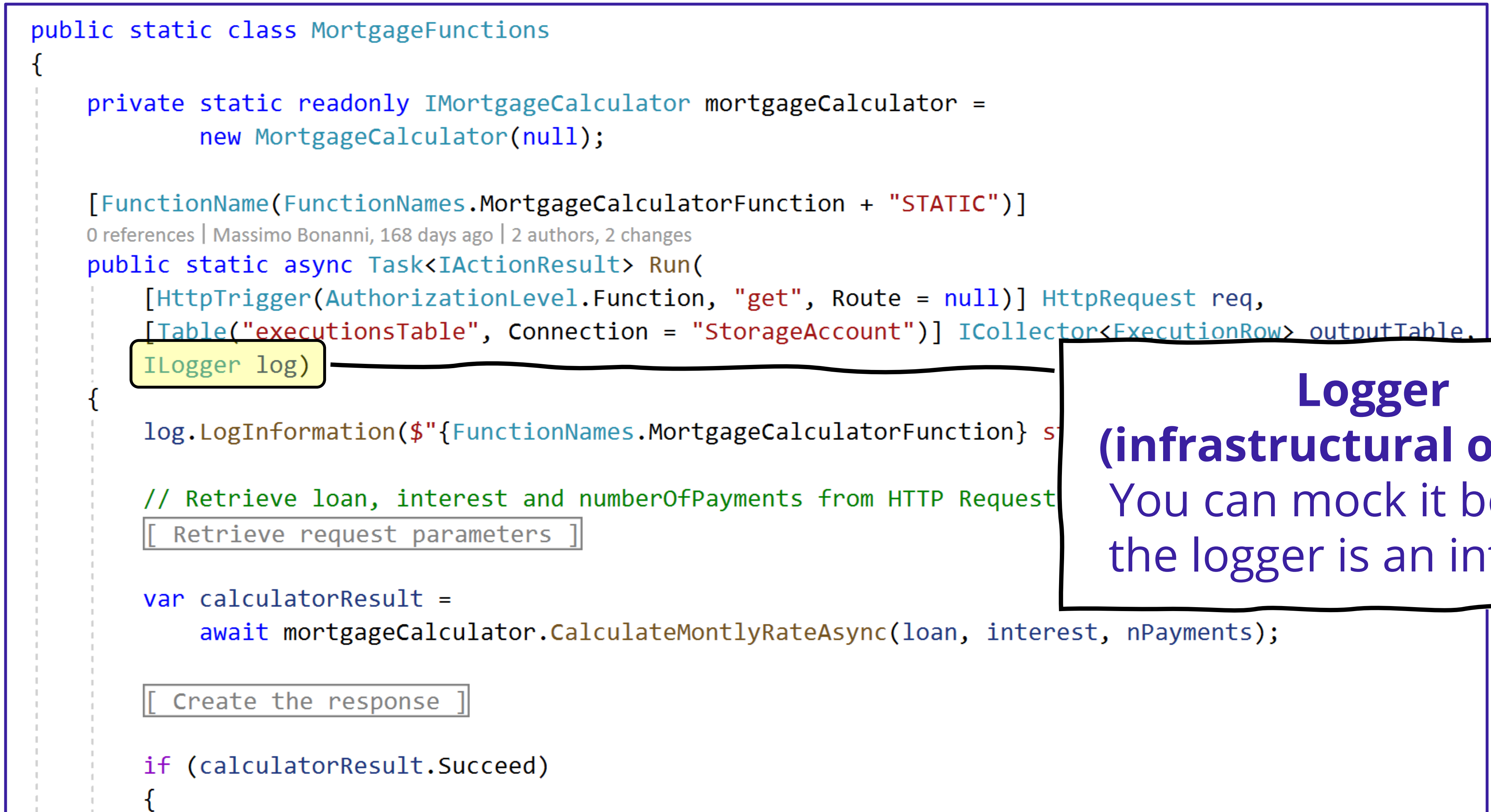
# Azure Function … bindings!!



```
public static class MortgageFunctions
{
    private static readonly IMortgageCalculator mortgageCalculator =
        new MortgageCalculator(null);

    [FunctionName(FunctionNames.MortgageCalculatorFunction + "STATIC")]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 2 changes
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} start");

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await mortgageCalculator.CalculateMontlyRateAsync(loan, interest, nPay

        [ Create the response ]

        if (calculatorResult.Succeed)
        {
```

**Binding**
You can mock it because the binding payload is an interface

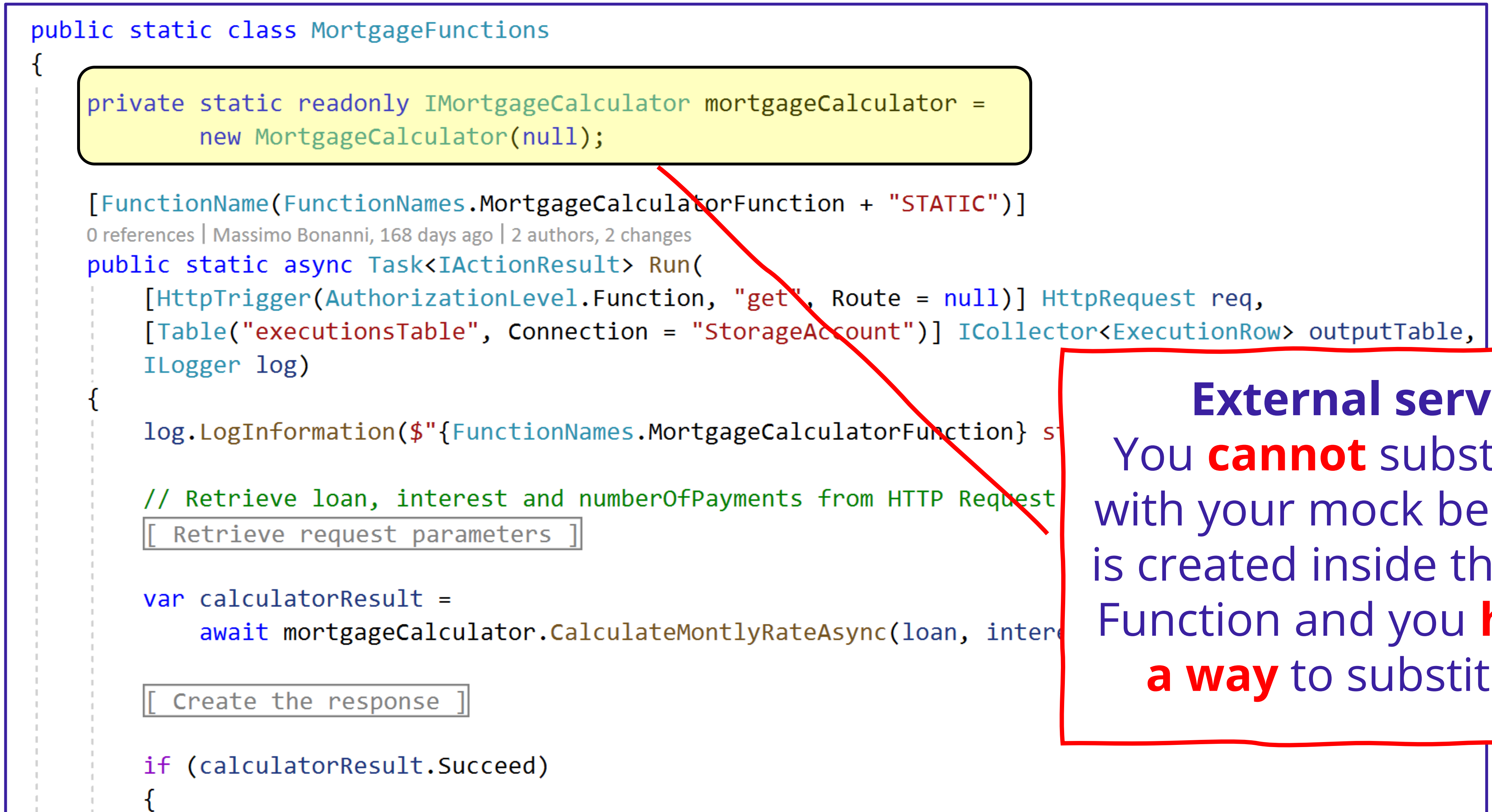
# Azure Function … logger!!



```
public static class MortgageFunctions
{
    private static readonly IMortgageCalculator mortgageCalculator =
            new MortgageCalculator(null);

    [FunctionName(FunctionNames.MortgageCalculatorFunction + "STATIC")]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 2 changes
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} s

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await mortgageCalculator.CalculateMontlyRateAsync(loan, interest, nPayments);

        [ Create the response ]

        if (calculatorResult.Succeed)
        {
```

**Logger**
**(infrastructural objects)**
You can mock it because the logger is an interface

# Azure Function … your service!!



```
public static class MortgageFunctions
{
    private static readonly IMortgageCalculator mortgageCalculator =
            new MortgageCalculator(null);

    [FunctionName(FunctionNames.MortgageCalculatorFunction + "STATIC")]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 2 changes
    public static async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} s

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await mortgageCalculator.CalculateMontlyRateAsync(loan, inter

        [ Create the response ]

        if (calculatorResult.Succeed)
        {
```

**External service**

You **cannot** substitute it with your mock because it is created inside the Azure Function and you **haven't a way** to substitute it

# Make your Azure Function testable!!!

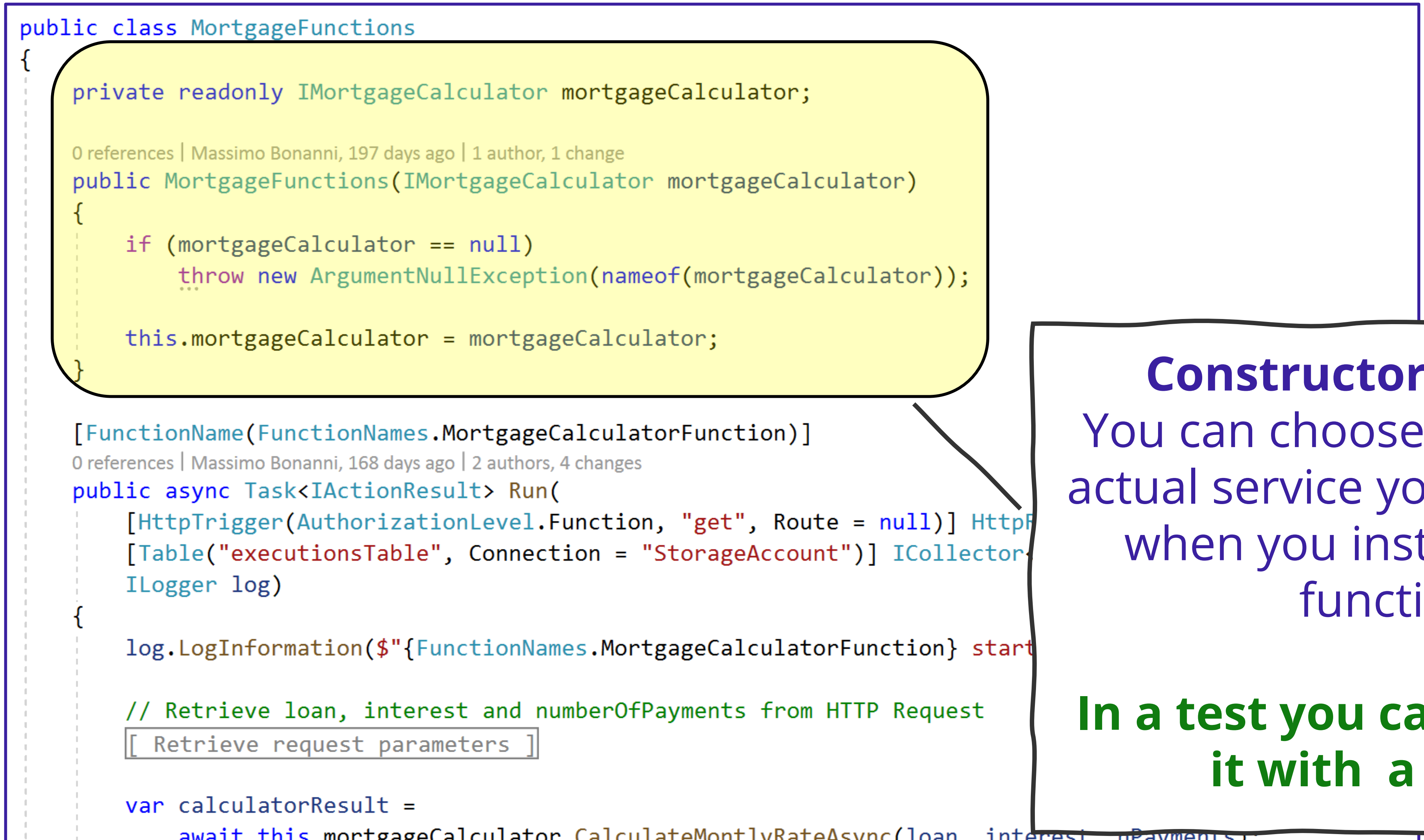
The solution of your problem is:
**Dependency Injection** !!

Azure Functions Runtime is based on .NET Core.

Azure Functions support the same ASP.NET Core Dependency Injection!!!

Using Dependency Injection you provide a way to substitute your Services with a mock!

# Azure Function … testable!!

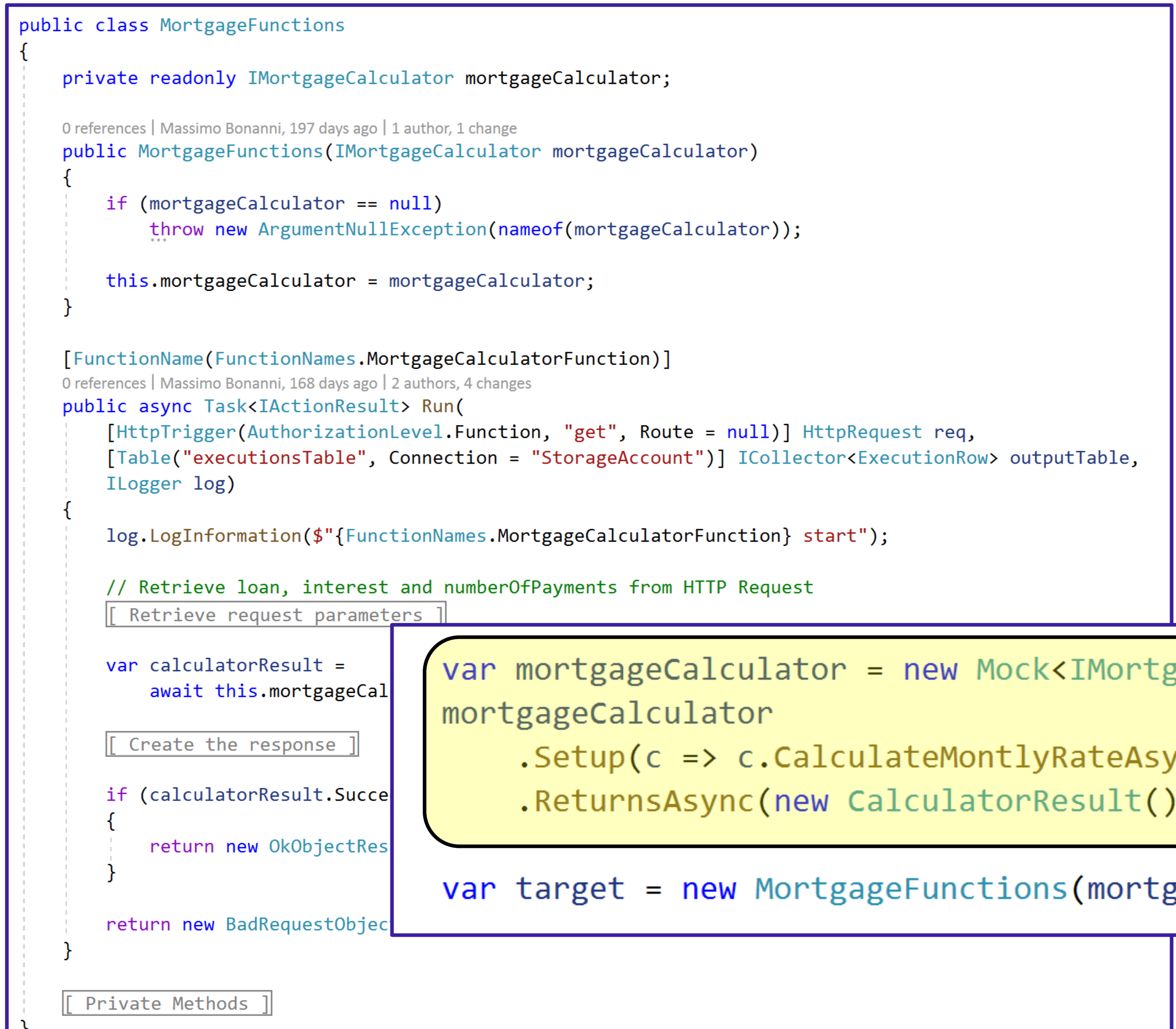


```
public class MortgageFunctions
{
    private readonly IMortgageCalculator mortgageCalculator;

    0 references | Massimo Bonanni, 197 days ago | 1 author, 1 change
    public MortgageFunctions(IMortgageCalculator mortgageCalculator)
    {
        if (mortgageCalculator == null)
            throw new ArgumentNullException(nameof(mortgageCalculator));

        this.mortgageCalculator = mortgageCalculator;
    }

    [FunctionName(FunctionNames.MortgageCalculatorFunction)]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 4 changes
    public async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpR
        [Table("executionsTable", Connection = "StorageAccount")] ICollector
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} start

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await this.mortgageCalculator.CalculateMontlyRateAsync(loan, interest, nPayments);
```

**Constructor Injection**
You can choose what kind of actual service you want to use when you instantiate the function.

**In a test you can substitute it with a mock!!**

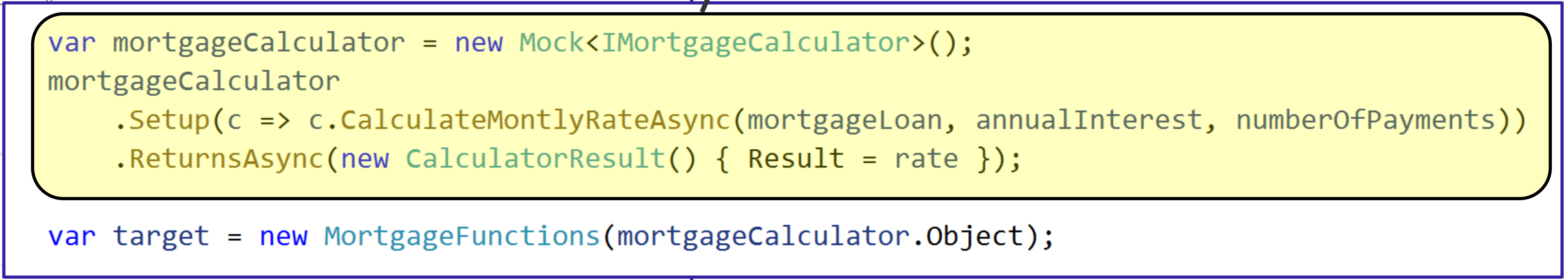
# Azure Function … how to use mock!!



```
public class MortgageFunctions
{
    private readonly IMortgageCalculator mortgageCalculator;

    0 references | Massimo Bonanni, 197 days ago | 1 author, 1 change
    public MortgageFunctions(IMortgageCalculator mortgageCalculator)
    {
        if (mortgageCalculator == null)
            throw new ArgumentNullException(nameof(mortgageCalculator));

        this.mortgageCalculator = mortgageCalculator;
    }

    [FunctionName(FunctionNames.MortgageCalculatorFunction)]
    0 references | Massimo Bonanni, 168 days ago | 2 authors, 4 changes
    public async Task<IActionResult> Run(
        [HttpTrigger(AuthorizationLevel.Function, "get", Route = null)] HttpRequest req,
        [Table("executionsTable", Connection = "StorageAccount")] ICollector<ExecutionRow> outputTable,
        ILogger log)
    {
        log.LogInformation($"{FunctionNames.MortgageCalculatorFunction} start");

        // Retrieve loan, interest and numberOfPayments from HTTP Request
        [ Retrieve request parameters ]

        var calculatorResult =
            await this.mortgageCal

        [ Create the response ]

        if (calculatorResult.Succe
        {
            return new OkObjectRes
        }

        return new BadRequestObjec
    }

    [ Private Methods ]
}
```

**Mock**

Create a mock to use in the test!!



```
var mortgageCalculator = new Mock<IMortgageCalculator>();
mortgageCalculator
    .Setup(c => c.CalculateMontlyRateAsync(mortgageLoan, annualInterest, numberOfPayments))
    .ReturnsAsync(new CalculatorResult() { Result = rate });

var target = new MortgageFunctions(mortgageCalculator.Object);
```

# DEMO

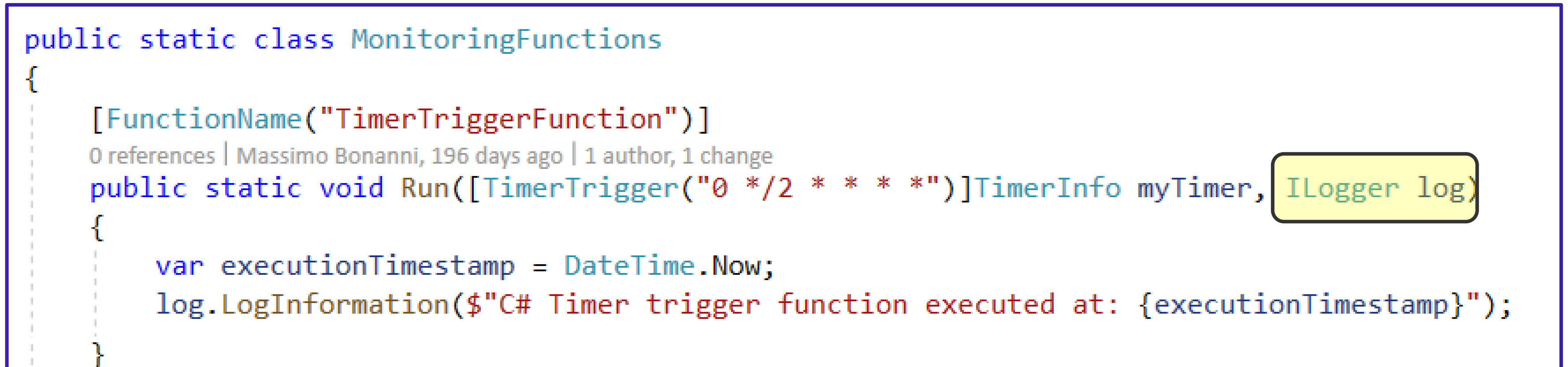
Azure Functions
Unit Testing

# Monitoring Azure Functions

Once you deploy your Azure Functions on Azure, you need to monitor them to check when something goes wrong.

The signature of an Azure Function Run method provides the instance of **ILogger** that you can use to log information about your code.

Using **ILogger**, you can collect information from your code execution to monitor and triage errors and exceptions.

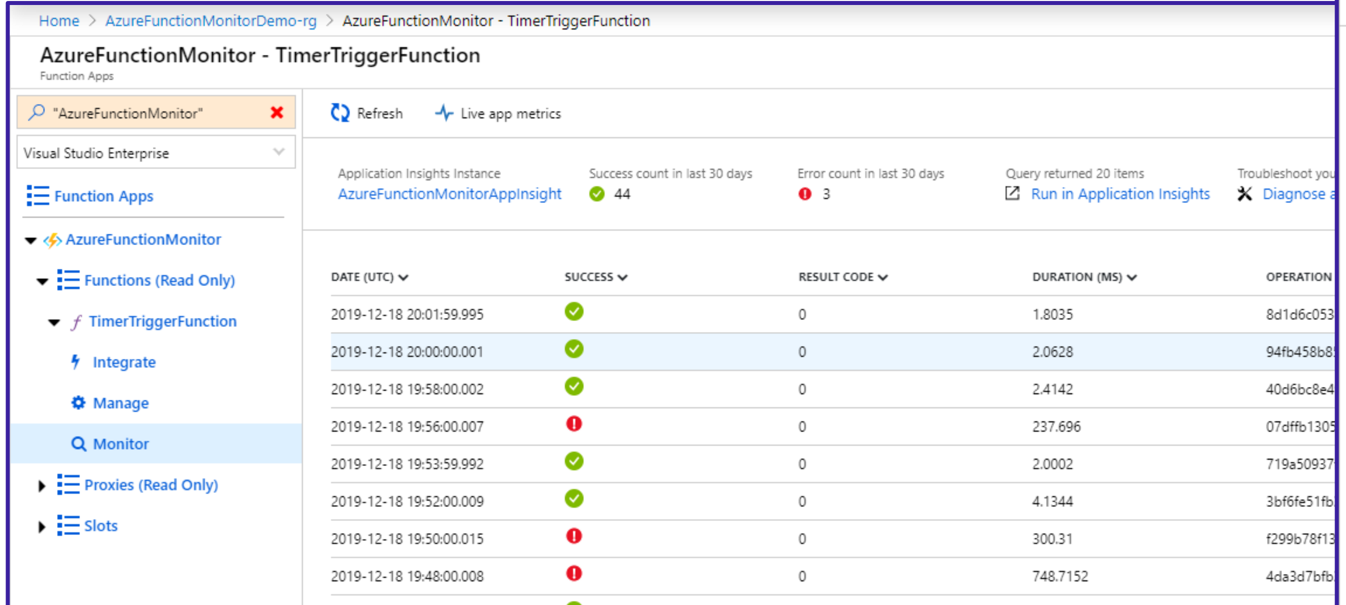


```
public static class MonitoringFunctions
{
    [FunctionName("TimerTriggerFunction")]
    0 references | Massimo Bonanni, 196 days ago | 1 author, 1 change
    public static void Run([TimerTrigger("0 */2 * * * *")]TimerInfo myTimer, ILogger log)
    {
        var executionTimestamp = DateTime.Now;
        log.LogInformation($"C# Timer trigger function executed at: {executionTimestamp}");
    }
```

# Azure Functions Monitor

Azure Functions provide out-of-the-box monitor feature.

For each Function, you can have info about every function execution.



Home > AzureFunctionMonitorDemo-rg > AzureFunctionMonitor - TimerTriggerFunction

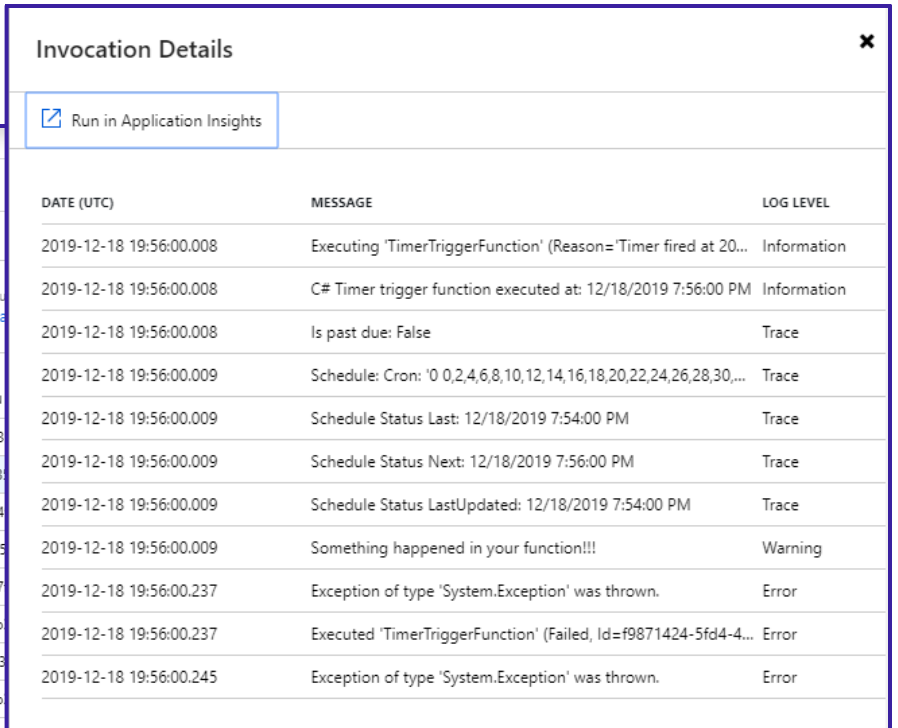
AzureFunctionMonitor - TimerTriggerFunction
Function Apps

"AzureFunctionMonitor"
Visual Studio Enterprise
Function Apps
AzureFunctionMonitor
Functions (Read Only)
TimerTriggerFunction
Integrate
Manage
Monitor
Proxies (Read Only)
Slots

Refresh | Live app metrics

Application Insights Instance: AzureFunctionMonitorAppInsight
Success count in last 30 days: ✓ 44
Error count in last 30 days: ❗ 3
Query returned 20 items: Run in Application Insights
Troubleshoot you: Diagnose a

| DATE (UTC) | SUCCESS | RESULT CODE | DURATION (MS) | OPERATION |
|---|---|---|---|---|
| 2019-12-18 20:01:59.995 | ✓ | 0 | 1.8035 | 8d1d6c053 |
| 2019-12-18 20:00:00.001 | ✓ | 0 | 2.0628 | 94fb458b8 |
| 2019-12-18 19:58:00.002 | ✓ | 0 | 2.4142 | 40d6bc8e4 |
| 2019-12-18 19:56:00.007 | ❗ | 0 | 237.696 | 07dffb1305 |
| 2019-12-18 19:53:59.992 | ✓ | 0 | 2.0002 | 719a50937 |
| 2019-12-18 19:52:00.009 | ✓ | 0 | 4.1344 | 3bf6fe51fb |
| 2019-12-18 19:50:00.015 | ❗ | 0 | 300.31 | f299b78f13 |
| 2019-12-18 19:48:00.008 | ❗ | 0 | 748.7152 | 4da3d7bfb |



Invocation Details

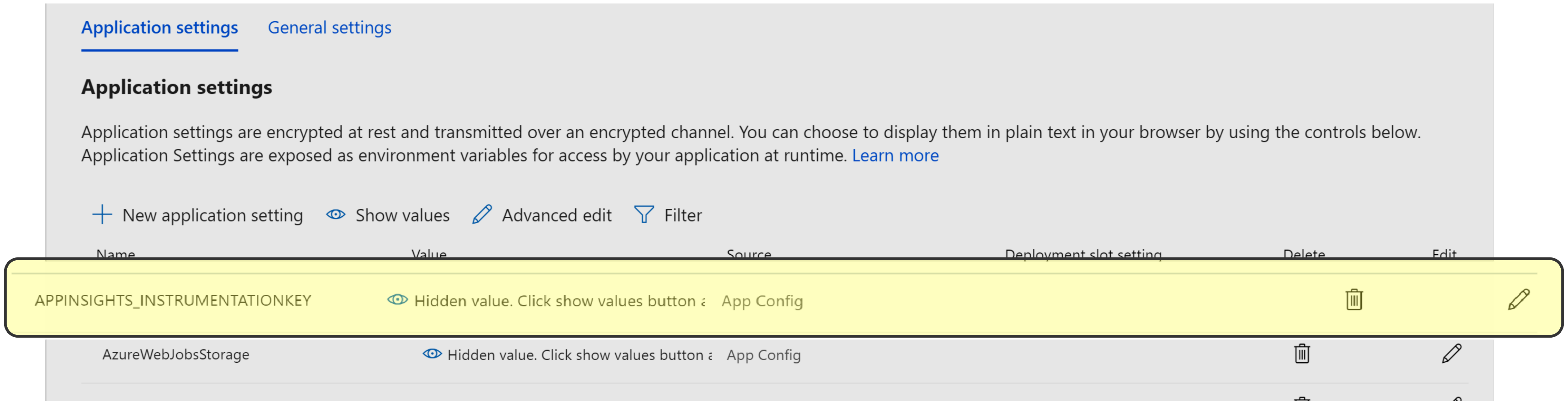
Run in Application Insights

| DATE (UTC) | MESSAGE | LOG LEVEL |
|---|---|---|
| 2019-12-18 19:56:00.008 | Executing 'TimerTriggerFunction' (Reason='Timer fired at 20... | Information |
| 2019-12-18 19:56:00.008 | C# Timer trigger function executed at: 12/18/2019 7:56:00 PM | Information |
| 2019-12-18 19:56:00.008 | Is past due: False | Trace |
| 2019-12-18 19:56:00.009 | Schedule: Cron: '0 0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,... | Trace |
| 2019-12-18 19:56:00.009 | Schedule Status Last: 12/18/2019 7:54:00 PM | Trace |
| 2019-12-18 19:56:00.009 | Schedule Status Next: 12/18/2019 7:56:00 PM | Trace |
| 2019-12-18 19:56:00.009 | Schedule Status LastUpdated: 12/18/2019 7:54:00 PM | Trace |
| 2019-12-18 19:56:00.009 | Something happened in your function!!! | Warning |
| 2019-12-18 19:56:00.237 | Exception of type 'System.Exception' was thrown. | Error |
| 2019-12-18 19:56:00.237 | Executed 'TimerTriggerFunction' (Failed, Id=f9871424-5fd4-4... | Error |
| 2019-12-18 19:56:00.245 | Exception of type 'System.Exception' was thrown. | Error |

# Azure Functions and Application Insight

The Azure Functions platform offers built-in integration with Azure Application Insights.

Put the **Application Insights instrumentation key** in the function app settings.



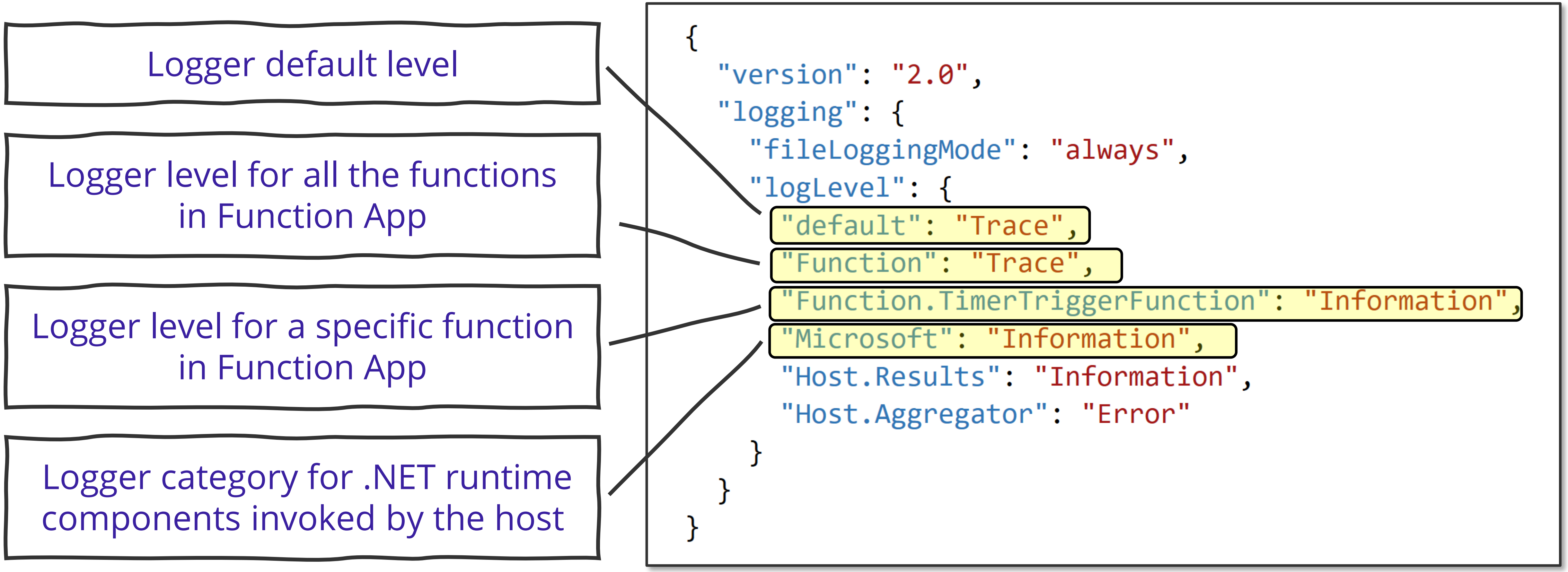
**Application settings**

Application settings are encrypted at rest and transmitted over an encrypted channel. You can choose to display them in plain text in your browser by using the controls below. Application Settings are exposed as environment variables for access by your application at runtime. Learn more

| Name | Value | Source | Deployment slot setting | Delete | Edit |
|---|---|---|---|---|---|
| APPINSIGHTS_INSTRUMENTATIONKEY | Hidden value. Click show values button a | App Config | | | |
| AzureWebJobsStorage | Hidden value. Click show values button a | App Config | | | |

# Configure monitoring

Logging is configured in **`host.json`** file.

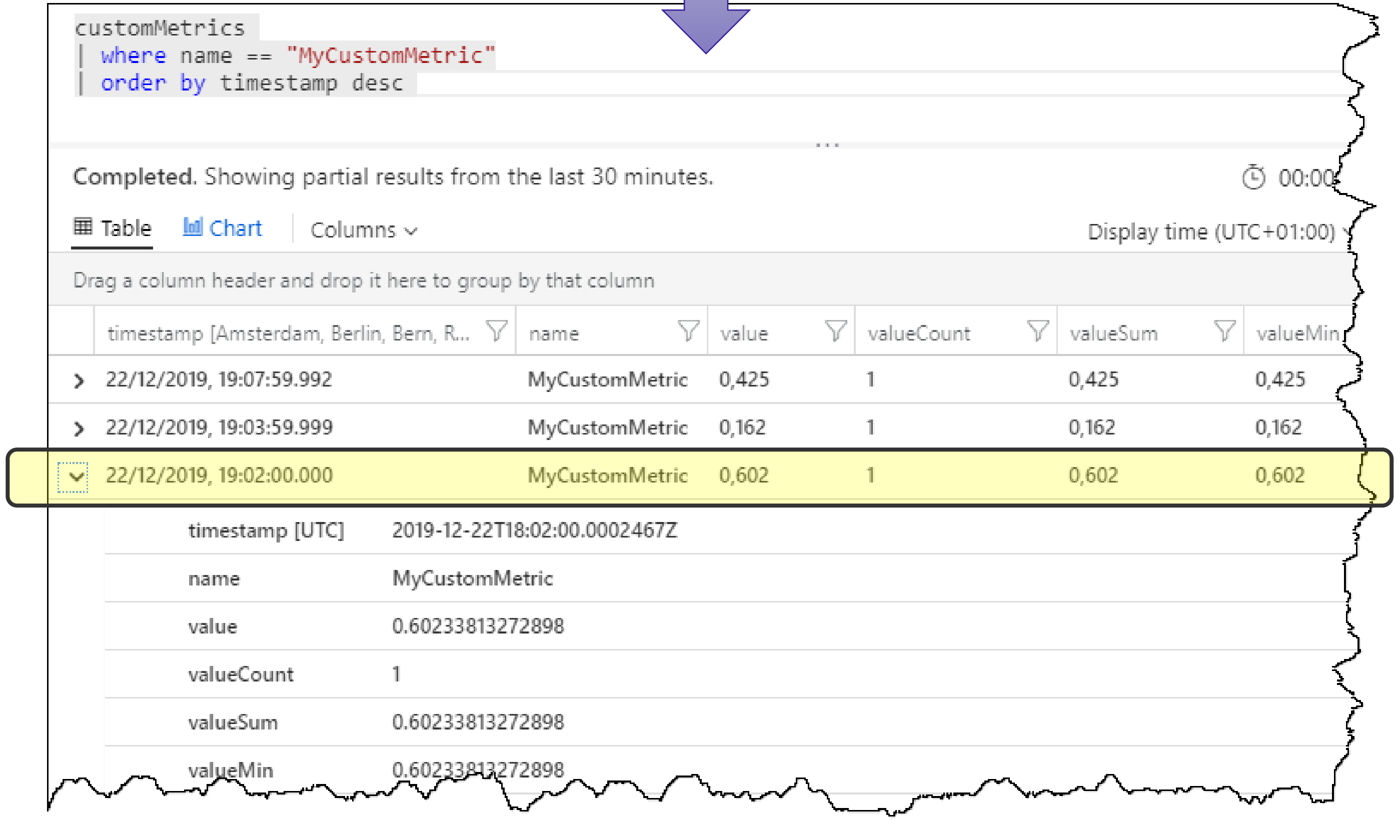

Logger default level
Logger level for all the functions in Function App
Logger level for a specific function in Function App
Logger category for .NET runtime components invoked by the host

```json
{
  "version": "2.0",
  "logging": {
    "fileLoggingMode": "always",
    "logLevel": {
      "default": "Trace",
      "Function": "Trace",
      "Function.TimerTriggerFunction": "Information",
      "Microsoft": "Information",
      "Host.Results": "Information",
      "Host.Aggregator": "Error"
    }
  }
}
```

# Custom Metrics

Azure Function SDK provides you extension methods to log custom metrics.

```
log.LogMetric("MyCustomMetric", CalculateMyCustomMetric());
```



```
customMetrics
| where name == "MyCustomMetric"
| order by timestamp desc
```

**Completed.** Showing partial results from the last 30 minutes. 00:00

Table Chart Columns Display time (UTC+01:00)

Drag a column header and drop it here to group by that column

| | timestamp [Amsterdam, Berlin, Bern, R... | name | value | valueCount | valueSum | valueMin |
|---|---|---|---|---|---|---|
| > | 22/12/2019, 19:07:59.992 | MyCustomMetric | 0,425 | 1 | 0,425 | 0,425 |
| > | 22/12/2019, 19:03:59.999 | MyCustomMetric | 0,162 | 1 | 0,162 | 0,162 |
| ⌄ | 22/12/2019, 19:02:00.000 | MyCustomMetric | 0,602 | 1 | 0,602 | 0,602 |

| | |
|---|---|
| timestamp [UTC] | 2019-12-22T18:02:00.0002467Z |
| name | MyCustomMetric |
| value | 0.60233813272898 |
| valueCount | 1 |
| valueSum | 0.60233813272898 |
| valueMin | 0.60233813272898 |

# DEMO

Azure Functions
Monitoring

# Takeaway

- Write an Azure Functions is **simple**!
- Testing Azure Functions is **simple**!
- Monitoring Azure Functions is **simple**!
- *.... then ....*

Photo by Aaron Burden on Unsplash

KEEP
CALM
AND
USE
AZURE FUNCTIONS

# Thanks for your attention!!!!!

**Massimo Bonanni**

*Azure Technical Trainer*

massimo.bonanni@microsoft.com
@massimobonanni
linkedin.com/in/massimobonanni/

Mastering
Azure Serverless Computing

A practical guide to build and deploy enterprise-grade serverless applications using Azure Functions

Lorenzo Barbieri and Massimo B

bit.ly/MasteringServerless

## References

- Azure Functions Documentation
  https://docs.microsoft.com/en-US/azure/azure-functions/
- Azure Functions Code Samples
  https://azure.microsoft.com/en-us/resources/samples/?service=functions&sort=0
- Azure Updates
  https://azure.microsoft.com/en-us/roadmap/?category=compute
- Azure Friday – Build Serverless APIs with Azure Functions
  https://azure.microsoft.com/en-us/resources/videos/azure-friday-build-serverless-apis-with-azure-functions/
- GitHub Demo
  https://github.com/massimobonanni/AzureFunctionsSamples

Photo by cottonbro from Pexels

DotNet Toscana

# Thanks for joining!

Ask questions on Twitter using #dotNETConf