Microsoft

Microsoft Ignite
The Tour

BRK30166
Empower every Azure Function to achieve more!!

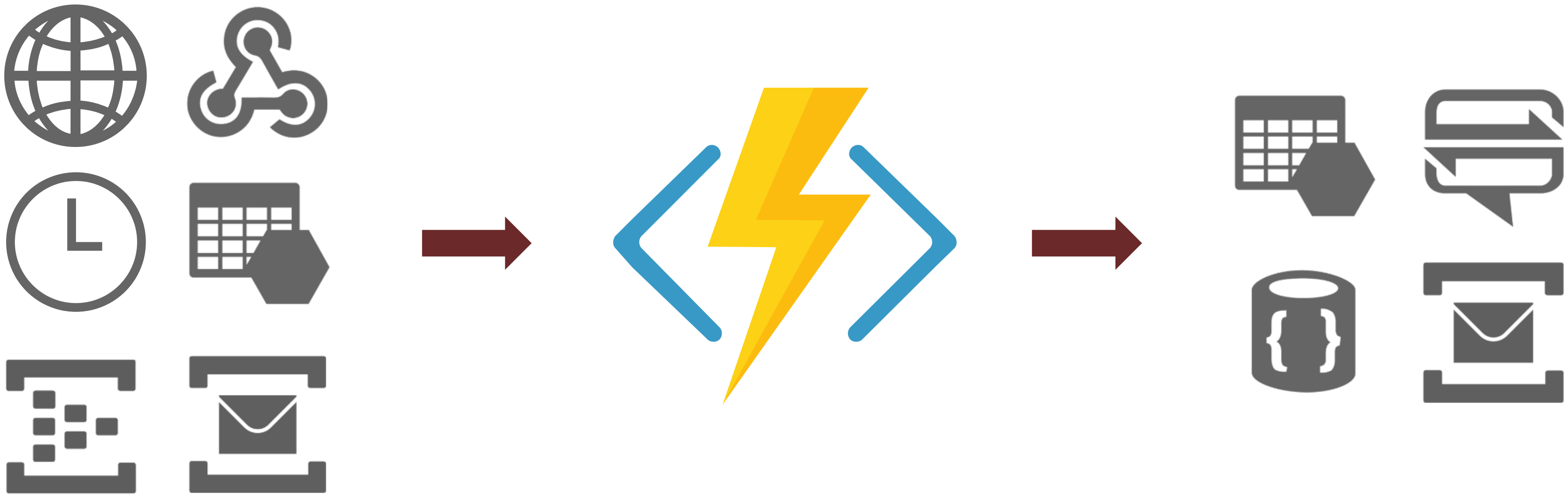
Massimo Bonanni

Paranormal Trainer, with the head in the Cloud and all the REST in microservices!

massimo.bonanni@microsoft.com

@massimobonanni

What are Azure Functions?



Events

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

Code

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

Outputs

Send results to an ever-growing collection of services

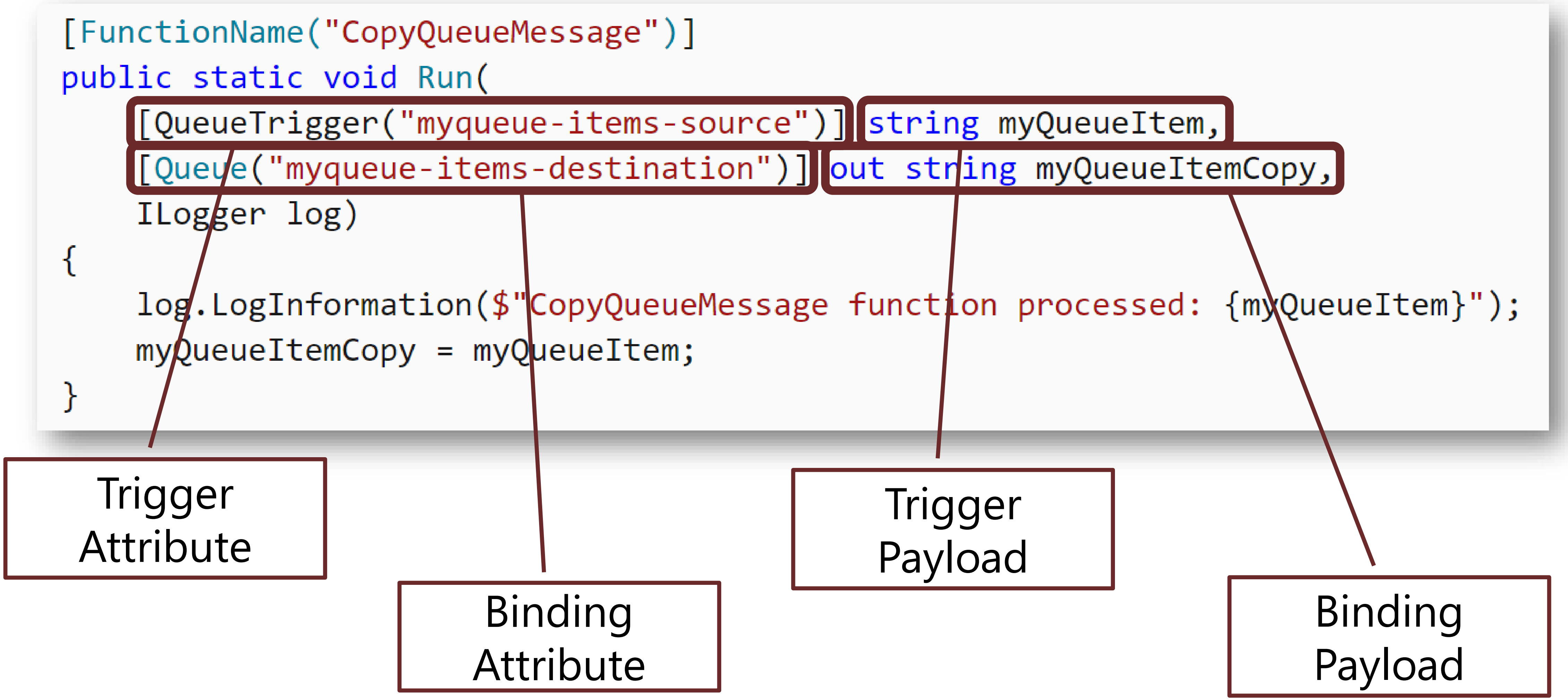
What can I do with Azure Functions?

Azure Functions is a great solution for processing data, integrating systems, working with the internet-of-things (IoT), and building simple APIs.

Azure Functions supports:

- Triggers, which are ways to start execution of your code (timer, http calls, item in a queue, …)
- Bindings, which are ways to simplify coding for input and output data (queue, blob storage, CosmosDB, SendGrid, …)

Anatomy of an Azure Function



```
[FunctionName("CopyQueueMessage")]
public static void Run(
    [QueueTrigger("myqueue-items-source")] string myQueueItem,
    [Queue("myqueue-items-destination")] out string myQueueItemCopy,
    ILogger log)
{
    log.LogInformation($"CopyQueueMessage function processed: {myQueueItem}");
    myQueueItemCopy = myQueueItem;
}
```

Trigger Attribute

Binding Attribute

Trigger Payload

Binding Payload

Extend triggers and bindings

All Triggers and Bindings (except for HTTPTrigger and Timer Trigger) are available as external packages (NuGet).

The Azure Functions SDK is based on the Azure WebJobs SDK and inherits the extension SDK from it.

An extension is a class that implements the IExtensionConfigProvider interface.

Azure Functions lifecycle phases

Startup

The runtime executes this phase only when the host starts.

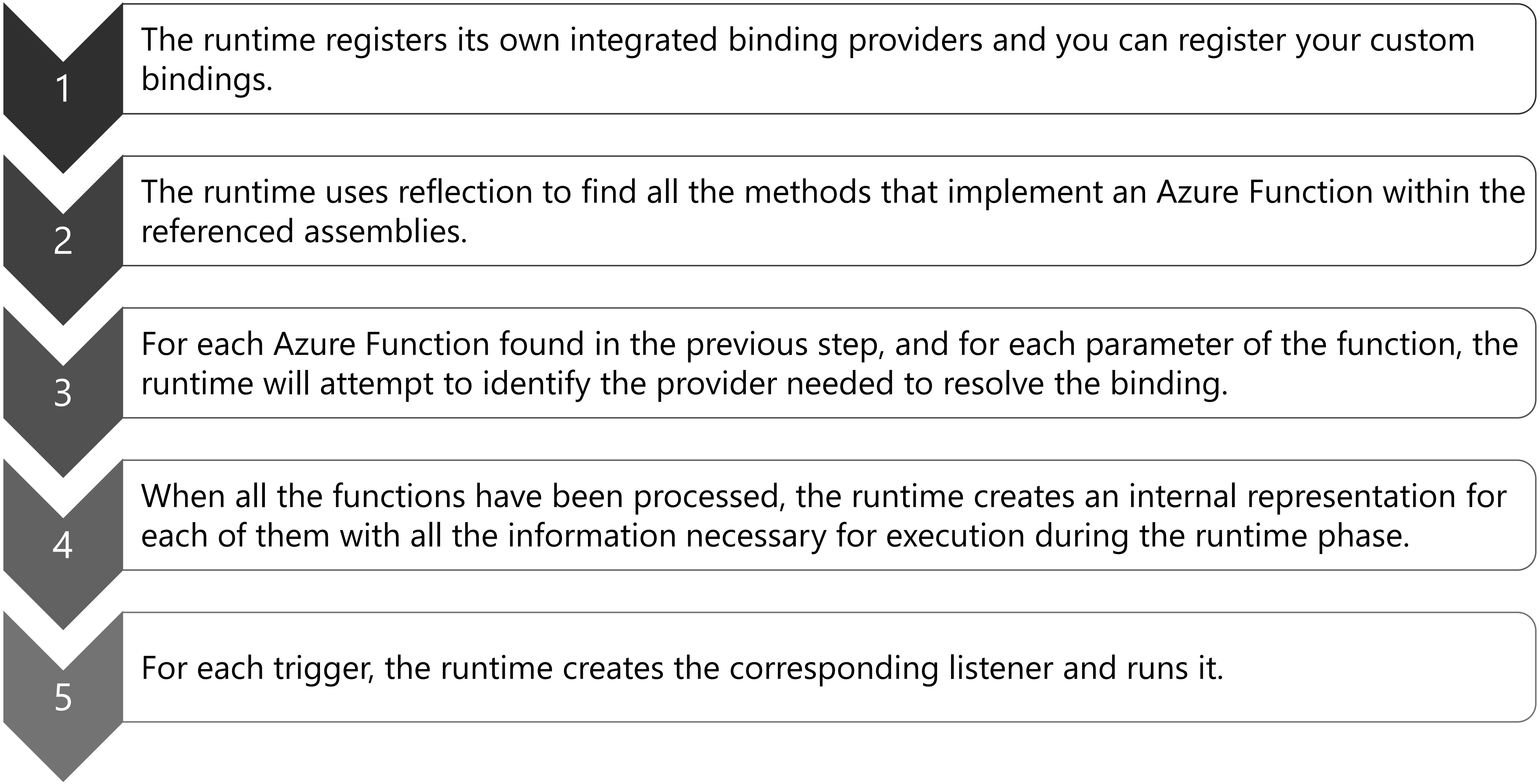
The runtime registers the built-in binding (TimerTrigger and HttpTrigger).

You must register your custom extensions.

Runtime

The runtime executes this phase every time a function is triggered by an event.

The Startup phase

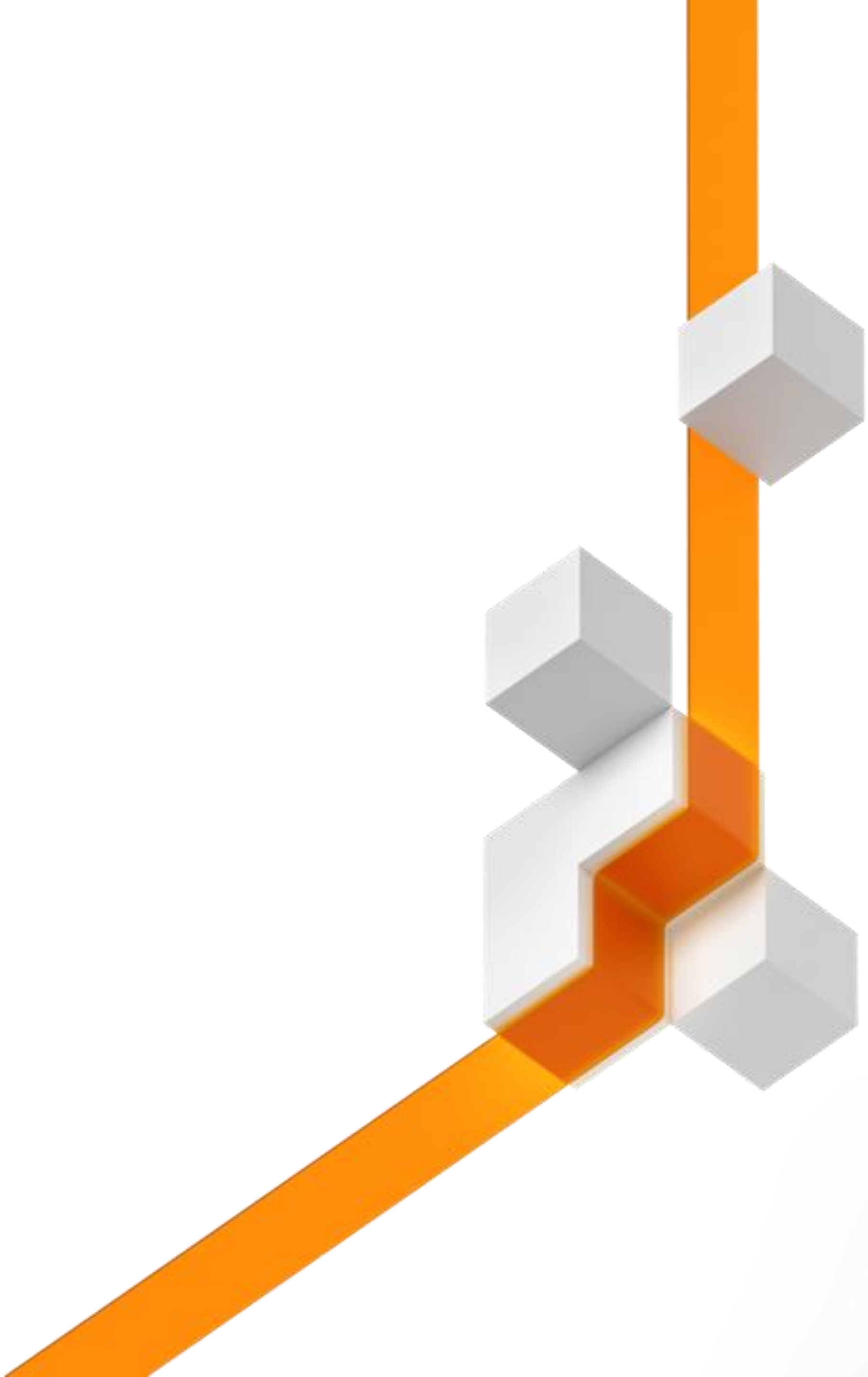


1. The runtime registers its own integrated binding providers and you can register your custom bindings.
2. The runtime uses reflection to find all the methods that implement an Azure Function within the referenced assemblies.
3. For each Azure Function found in the previous step, and for each parameter of the function, the runtime will attempt to identify the provider needed to resolve the binding.
4. When all the functions have been processed, the runtime creates an internal representation for each of them with all the information necessary for execution during the runtime phase.
5. For each trigger, the runtime creates the corresponding listener and runs it.

The Runtime phase

1. The runtime retrieves the complete definition of the function that it created during the startup phase and, for each binding, executes the **BindAsync** method (both for Trigger then Bindings).
2. The binding has the responsibility of converting the input values to the values actually used by the function.
3. If all binding methods are executed without any exceptions, the function is executed.

Trigger in deep



Classes involved in a Trigger

TriggerAttribute

TriggerConfigProvider

TriggerBindingProvider

TriggerBinding

TriggerListener

Classes involved in a Trigger

Decorates an argument of a method to identify the trigger.

TriggerAttribute

TriggerConfigProvider

TriggerBindingProvider

TriggerBinding

TriggerListener

Classes involved in a Trigger

Decorates an argument of a method to identify the trigger.

TriggerAttribute

Define the extension (implementing the IExtensionConfigProvider).

TriggerConfigProvider

TriggerBindingProvider

TriggerBinding

TriggerListener

Classes involved in a Trigger

Decorates an argument of a method to identify the trigger.

TriggerAttribute

Define the extension (implementing the IExtensionConfigProvider).

TriggerConfigProvider

Factory class for creating the actual binding object.

TriggerBindingProvider

TriggerBinding

TriggerListener

Classes involved in a Trigger

Decorates an argument of a method to identify the trigger.

TriggerAttribute

Define the extension (implementing the IExtensionConfigProvider).

TriggerConfigProvider

Factory class for creating the actual binding object.

TriggerBindingProvider

Binding object, creates the actual listener.

TriggerBinding

TriggerListener

Classes involved in a Trigger

Decorates an argument of a method to identify the trigger.

TriggerAttribute

Define the extension (implementing the IExtensionConfigProvider).

TriggerConfigProvider

Factory class for creating the actual binding object.

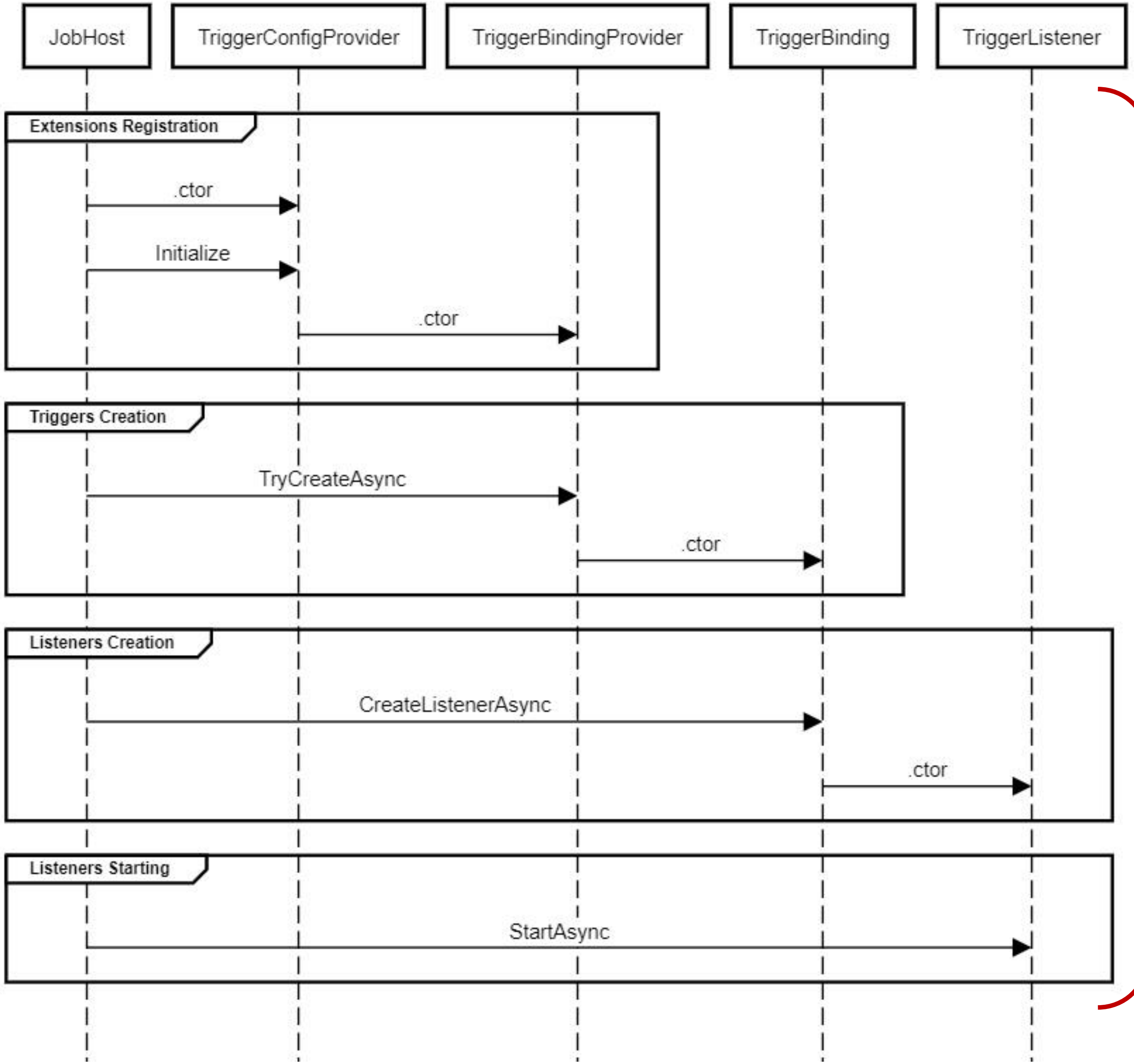
TriggerBindingProvider

Binding object, creates the actual listener.

TriggerBinding

It reacts to events and executing the function.

TriggerListener



The runtime creates the factory for binding

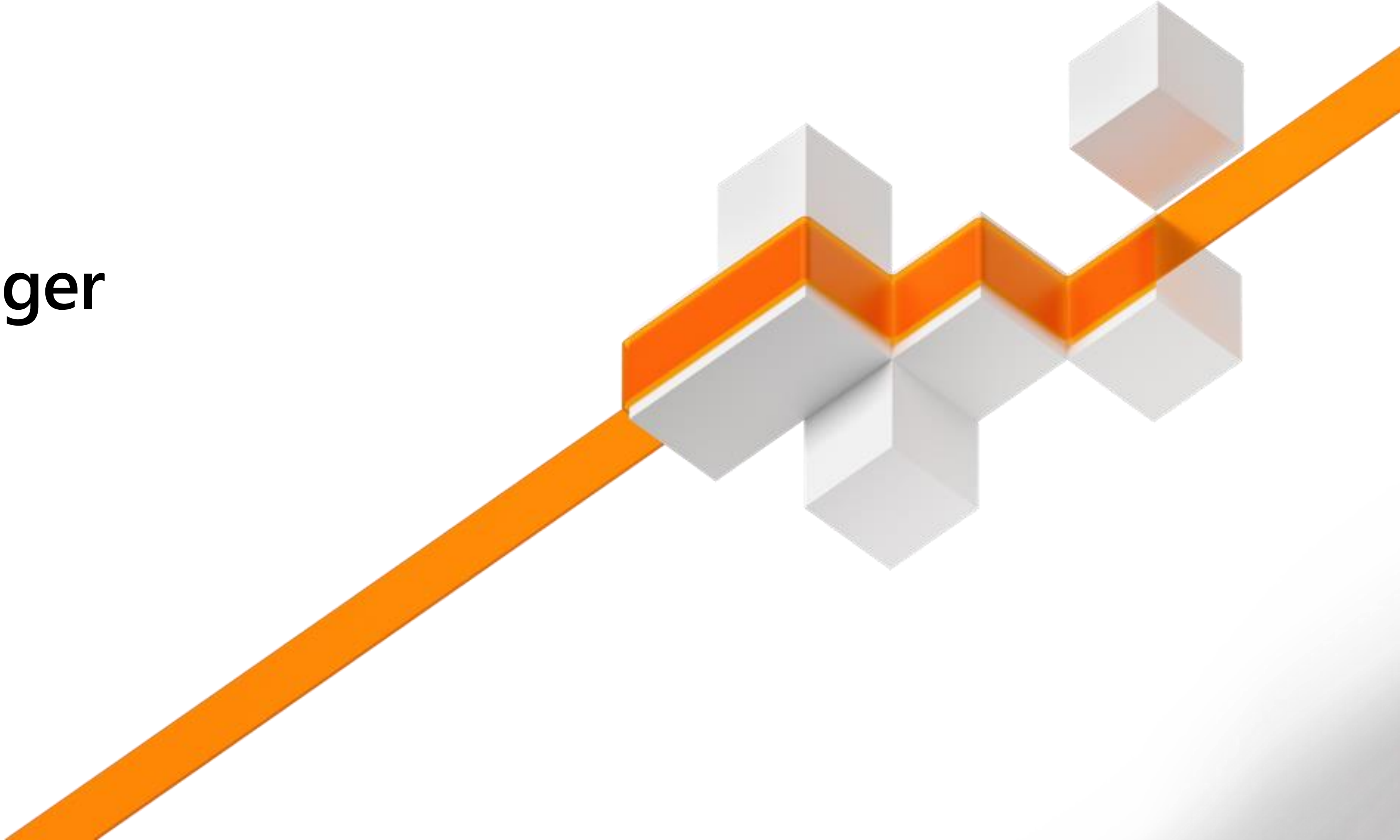
The runtime uses the factory to create the binding instance

The runtime uses binding class to create the listener

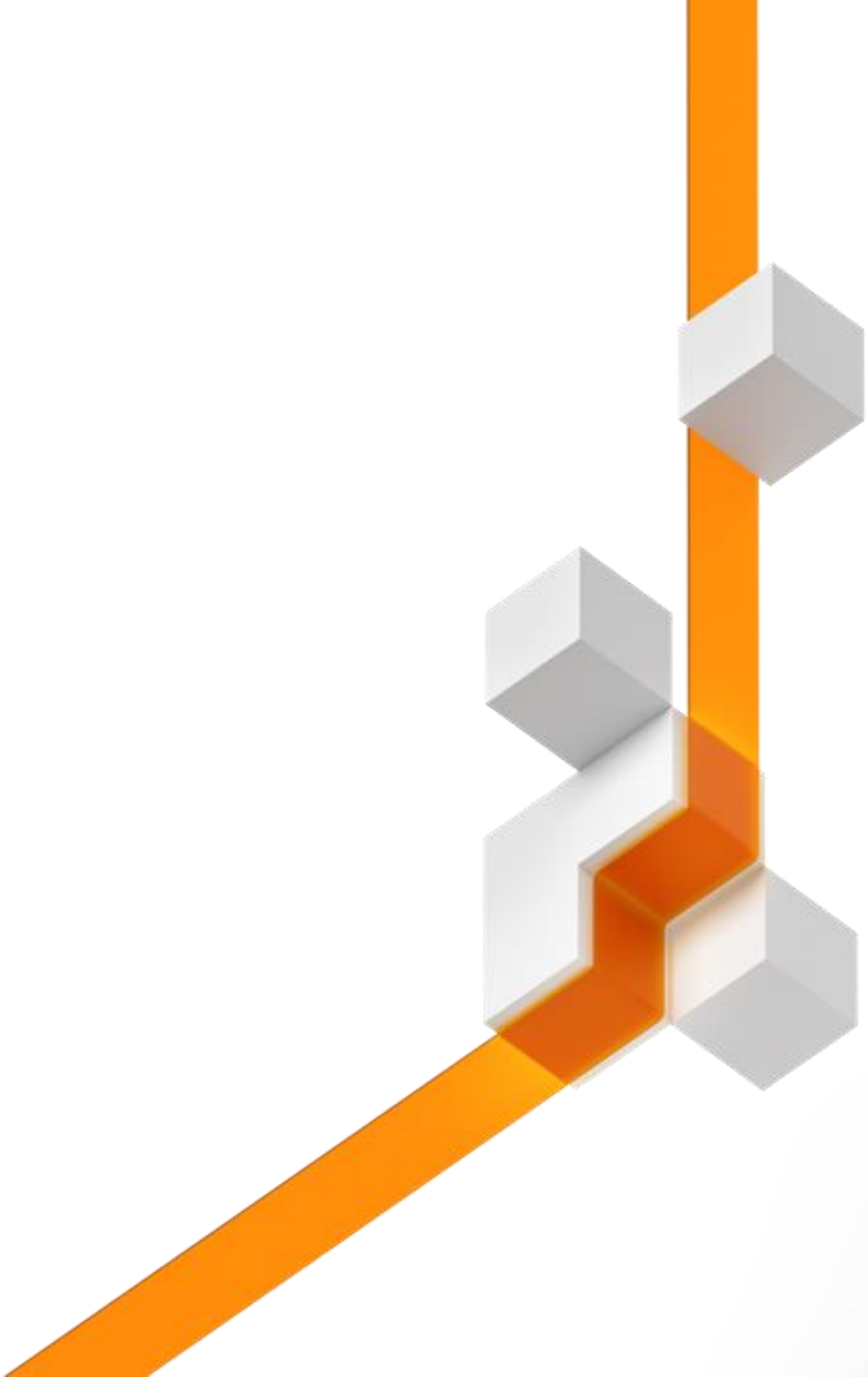
The runtime starts the listener

Startup Phase

DEMO
Weather Trigger



Binding in deep



Classes involved in a Binding

BindingAttribute

BindingConfigProvider

BindingConverter

Binding class

Classes involved in a Binding

BindingAttribute

Decorates an argument of a method to identify the binding.

BindingConfigProvider

BindingConverter

Binding class

Classes involved in a Binding

BindingAttribute

Decorates an argument of a method to identify the binding.

BindingConfigProvider

Define the extension (implementing the IExtensionConfigProvider).

BindingConverter

Binding class

Classes involved in a Binding

BindingAttribute

Decorates an argument of a method to identify the binding.

BindingConfigProvider

Define the extension (implementing the IExtensionConfigProvider).

BindingConverter

Creates the actual binding class for the binding.

Binding class

Classes involved in a Binding

Decorates an argument of a method to identify the binding.

BindingAttribute

Define the extension (implementing the IExtensionConfigProvider).

BindingConfigProvider

Creates the actual binding class for the binding.

BindingConverter

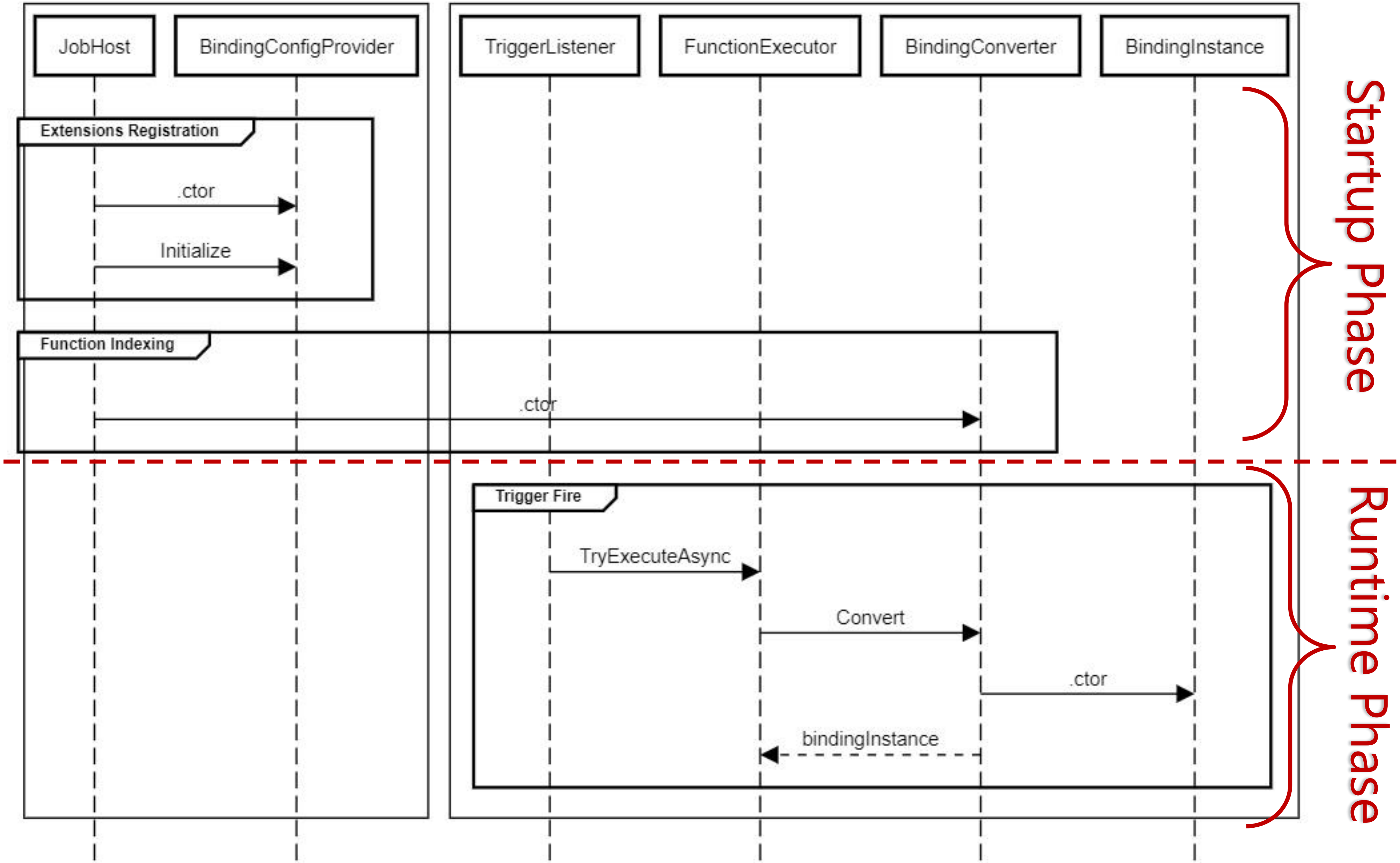
The class that actually binds to the data source

Binding class

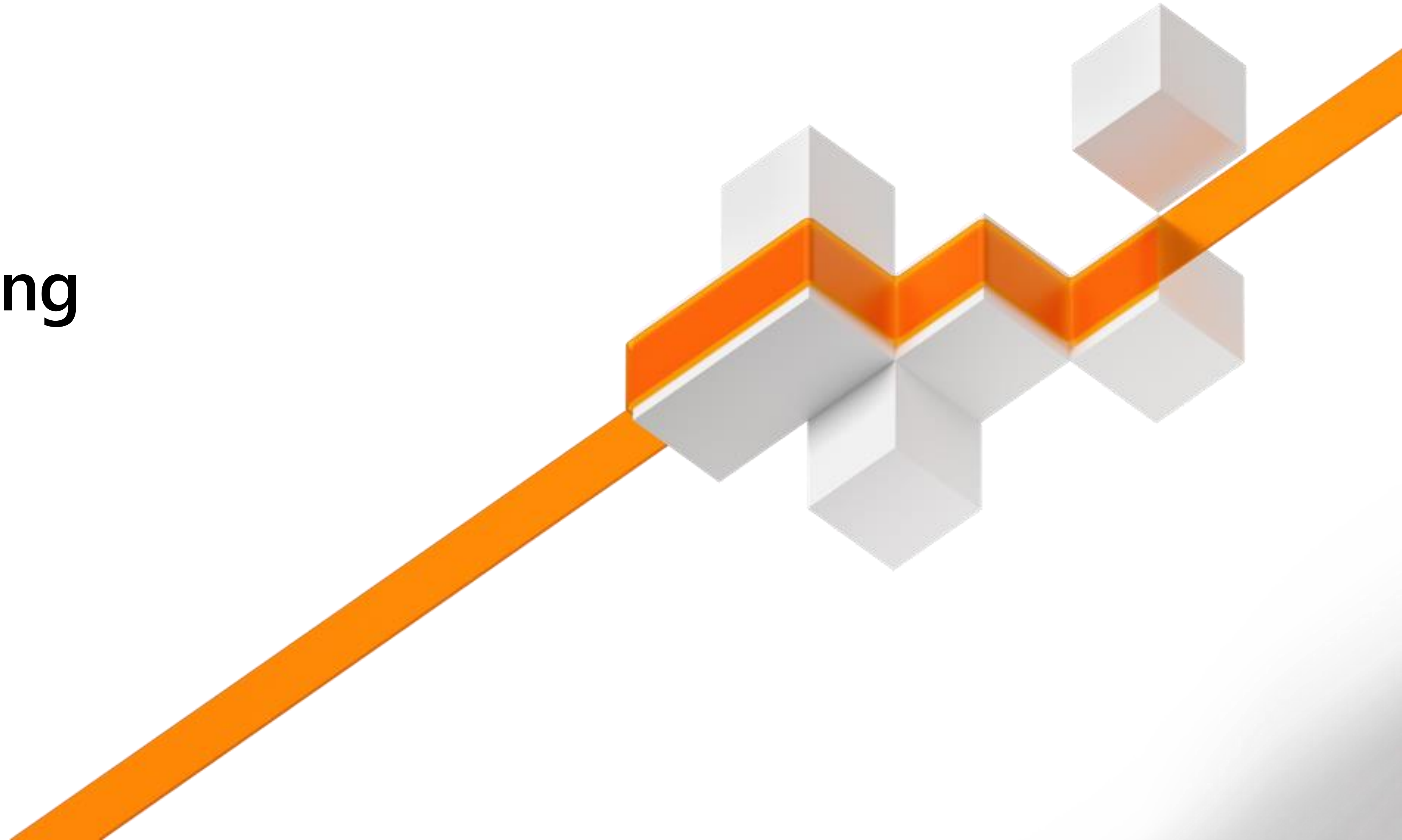
The runtime creates the extension.

The runtime creates the converter

When the listener fires a function (FunctionExecutor), the runtime uses the converter to create the binding class



DEMO
Twitter Binding



Conclusion

Implementing your own triggers and bindings allows you to abstract the data source with respect to the Azure Function code.

You pay for duration and memory occupation of your function. Your code must be efficiently and avoid to load assembly that you don’t use.

Trigger listener is one of the most important classes for scalability and performance: write it once and the best you can!!!

BRK30166

Thanks for your attention!!!!!

Q&A

Massimo Bonanni

Azure Technical Trainer @ Microsoft

massimo.bonanni@microsoft.com
@massimobonanni

Microsoft CERTIFIED AZURE FUNDAMENTALS

Microsoft CERTIFIED AZURE DEVELOPER ASSOCIATE

Microsoft CERTIFIED AZURE SOLUTIONS ARCHITECT EXPERT

Microsoft CERTIFIED AZURE DEVOPS ENGINEER EXPERT

MICROSOFT CERTIFIED TRAINER Microsoft MCT Trainer 2019 - 2020

TECHNOLOGY SOLUTION ACHIEVERS Microsoft Event Speaker Silver – FY19

TECHNOLOGY SOLUTION ACHIEVERS Microsoft External User Group Platinum – FY19

Connect with me on LinkedIn

linkedin.com/in/massimobonanni/

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

- Demo AccuWeather Trigger / Twitter Binding – GitHub

  http://bit.ly/AFCustomise

- Demo SQL Trigger/Binding – GitHub

  http://bit.ly/AFSQLTrigger

Microsoft