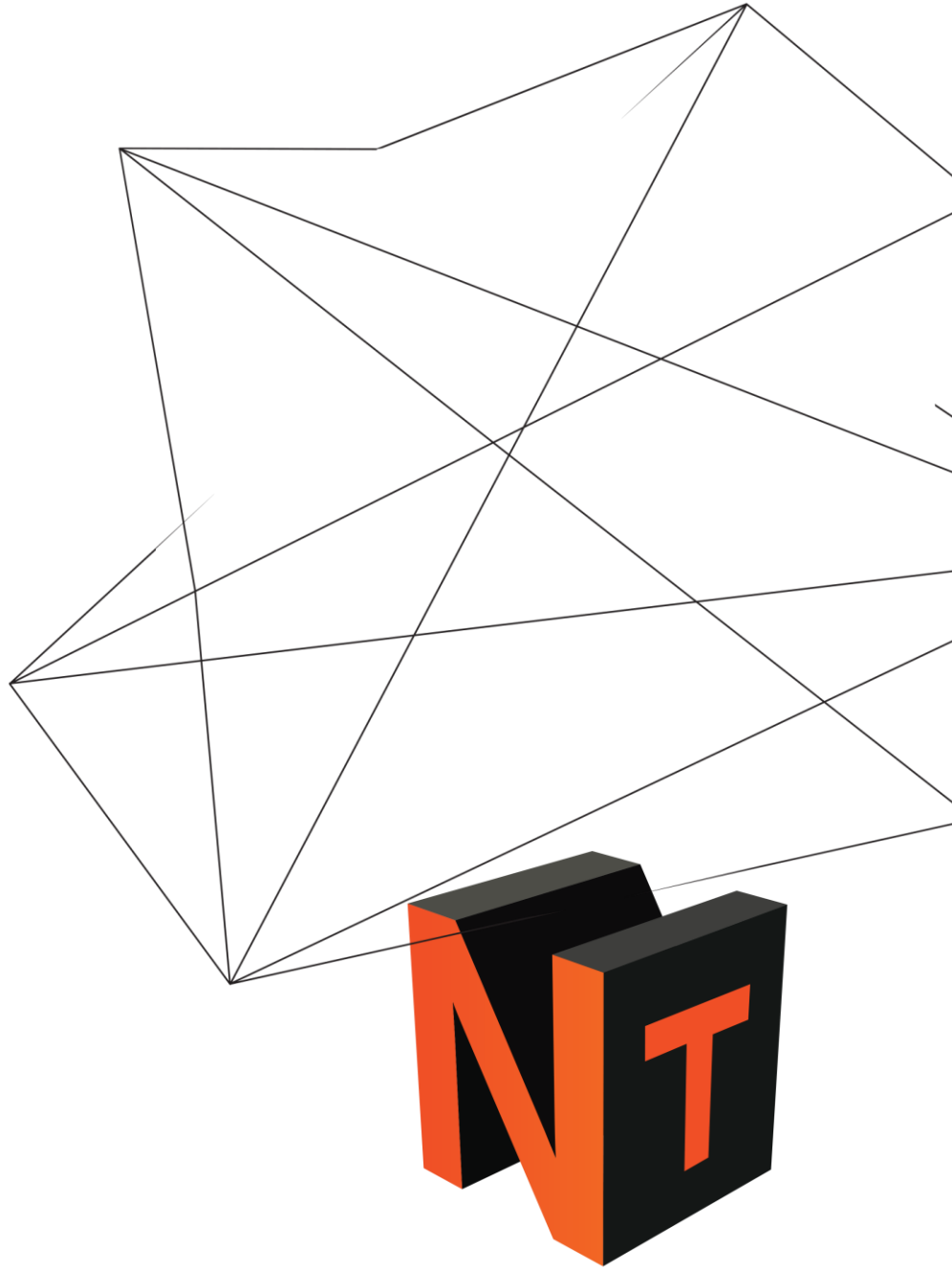
Microsoft

Azure Functions with Durable Functions Extension

Gregor Filej, Evizija d.o.o.
gregor@evizija.si



#ntk18

2018
NT Konferenca
Portorož | 22. - 24. maj 2018

Serverless & Microservices

- Serverless
  - Abstraction (IAAS -> PAAS -> Serverless)
  - scaling - event driven/instant scale
  - billing

- Microservices
  - loosely coupled,
  - collaborating services,
  - develop, test and deploy independently

Azure functions overview

- Runtime versions: 1.x - .NET Framework, 2.x .NET Core (pre-release)
- Billing -> consumption plan (gigabyte per second or # of executions) or AppService plan
- Scaling (scale controller) -> adding additional instances of function host -> instance of function host is Function App -> all functions within function app share resources within instance and scale at the same time
- Difference WebJobs vs Azure Functions (automatic scaling, pay-per-use)

Azure functions development

- Online development
  - Now available templates for Durable functions

- Local development
  - Azure functions core tools
  - Visual Studio 2017
  - Microsoft Azure Storage Emulator (local.appsettings.json)
  - Azure Functions and WebJob Tools extension
  - For Durable functions -> nuget Microsoft.Azure.Webjobs.Extensions.DurableTasks

Why use Durable Functions?

Simple workflow - as simple as it gets ☺

1.) WF within one function

2.) WF via messages

3.) WF via orchestrator function

Durable functions

- Azure Functions and the Durable Functions -> built on the WebJobs SDK

- **Orchestator f**. (define wf in code, call other activity functions, manages state, checkpoint progress (await!!!) and restarts)
- **Activity function**

Durable functions

Tables – checkpointing execution history with event sourcing pattern ->

- Instances
- History

Internal queue triggers for function invocation ->

- work item queue (activity f.)
- control queues (orchestrator f.)

Task Hub -> logical container for Azure storage resources ->

Orchestrator and activity functions can only interact with each other when they belong to the same task hub. Each function App has separate task hub. Storage account can have multiple task hubs.

Orchestrator function

HTTP API -> check progress, raise events, terminate

Error handling -> FunctionsFailedException, Automatic retry on failure

Timers -> delay, timeout

Diagnostics -> AppInsights, Logging, Custom Status, Debugging, Storage

Constraints non blocking, never initiate any async operation, infinite loops avoided (growing history)
– ContinueAsNew method (orchestrator function history truncated)

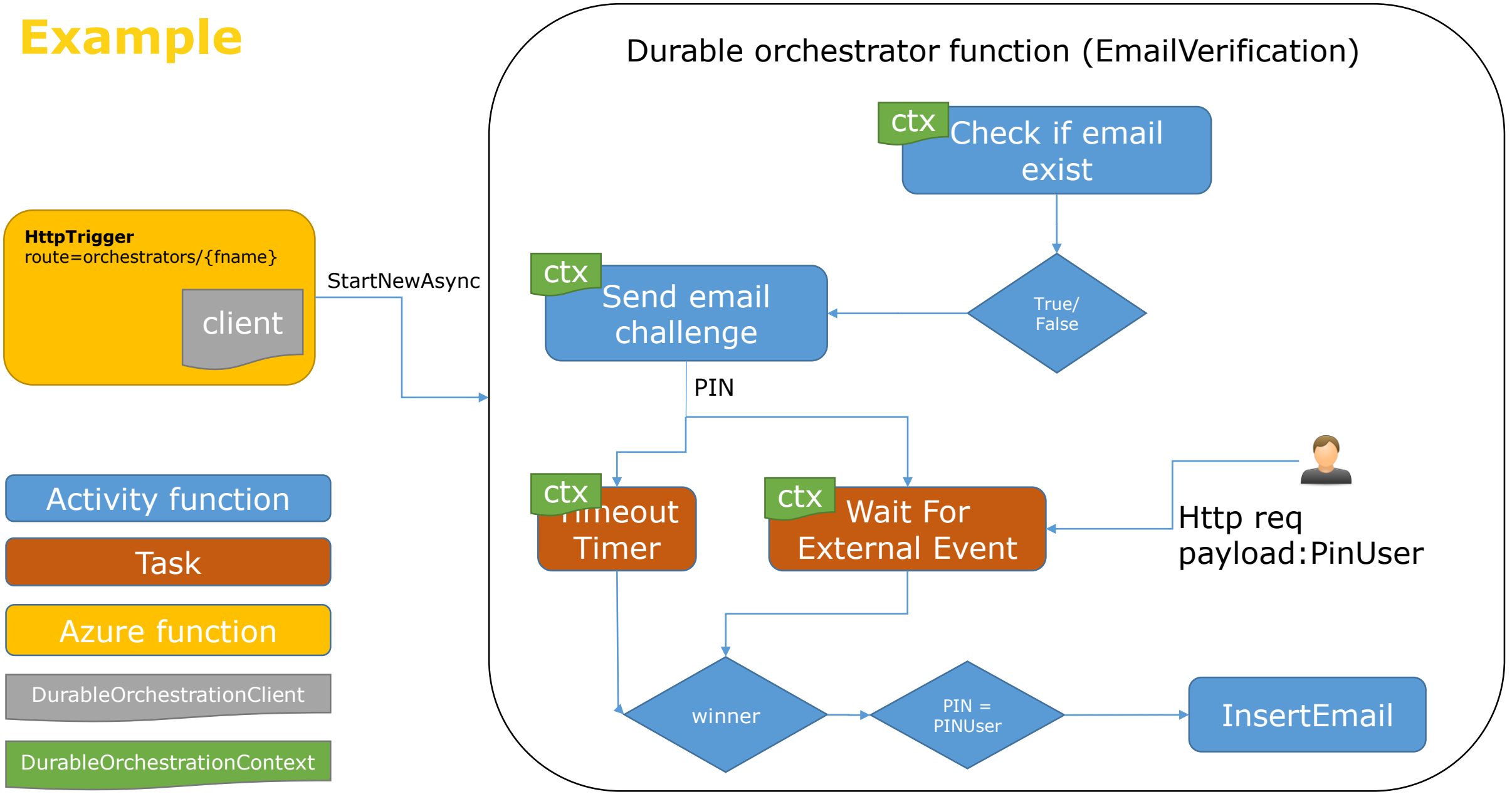
Activity function

- Stateless
- The same behaviors as regular queue-triggered functions.
- They can safely do I/O, execute CPU intensive operations, and use multiple threads.
- Because activity triggers are stateless, they can freely scale out to an unbounded number of VMs.

Patterns

- **Chaining**
- Fan-Out/Fan-In
- Async HTTP APIs
- Monitoring
- **Human Interaction**

Example


Durable orchestrator function (EmailVerification)
HttpTrigger
route=orchestrators/{fname}
client
StartNewAsync
ctx
Check if email exist
True/ False
ctx
Send email challenge
PIN
ctx
Timeout Timer
ctx
Wait For External Event
Http req payload:PinUser
winner
PIN = PINUser
InsertEmail
Activity function
Task
Azure function
DurableOrchestrationClient
DurableOrchestrationContext