Da un (Azure) Functions ad un'applicazione in 60 minuti

About me

@amelchiori

// melkio

// amelchiori

What does “serverless” mean?

- Reacts to events
- Autoscales
- Pay-as-you-go

Once upon a time...

...there were WebJobs...

What's a "WebJob"?

- WebJobs
  - Easily host background services in managed compute service, App Service
  - Continuously Running, Timed or on-demand
- WebJobs SDK
  - Function oriented programming model
  - **Trigger** on new events
  - Introduces **binding** concepts

What was the next step...

- Add language abstraction layer
- Provide a more lightweight language experience
- Only deploy functions. No hosts
- Platform handles execution/scale for developers
    - App Service already has auto-scale, but it involves decisions on users' part

How to create a “Function App”

- Portal
- CLI
  - ```
    az functionapp create \
    --resource-group myResourceGroup \
    --consumption-plan-location westeurope \
    --name <app_name> \
    --storage-account <storage_name>
    ```
    ```

Scale & hosting

- Consumption plan
  - instances of the Azure Functions host are dynamically added and removed based on the number of incoming events
  - don't have to pay for idle VMs and don't have to reserve capacity in advance
  - billing is based on number of executions, execution time, and memory used. Billing is aggregated across all functions within a function app
  - includes a monthly free grant of 1 million requests and 400,000 GB-s of resource consumption per month
- App Service plan
  - function apps run on dedicated VMs on Basic, Standard, Premium, and Isolated SKUs
  - VM decouples cost from number of executions, execution time, and memory used

Runtime versions

- 1.x:
  - supports development and hosting only in the portal or on Windows
- 2.x:
  - runs on .NET Core
  - can run on all platforms supported by .NET Core, including macOS and Linux
  - enables cross-platform development and hosting scenarios

How to target different runtime versions

- Portal
- Application settings
  - `FUNCTIONS_EXTENSION_VERSION = ~1 / ~2`
- CLI
  - ```
    az functionapp config appsettings set --name <function_app> \
    --resource-group <my_resource_group> \
    --settings FUNCTIONS_EXTENSION_VERSION=<version>
    ```
    ```

Trigger & binding

- A trigger defines how a function is invoked
  - A function must have exactly one trigger
  - Triggers have associated data
- Input and output bindings provide a declarative way to connect to data from within your code
  - bindings are optional and a function can have multiple input and output bindings
  - in 2.x, all bindings except HTTP, Timer, and Azure Storage must be registered
    - `func extensions install -p <package_name> -v <version>`
  - extensions are delivered as NuGet packages (microsoft.azure.webjobs.extensions.*)

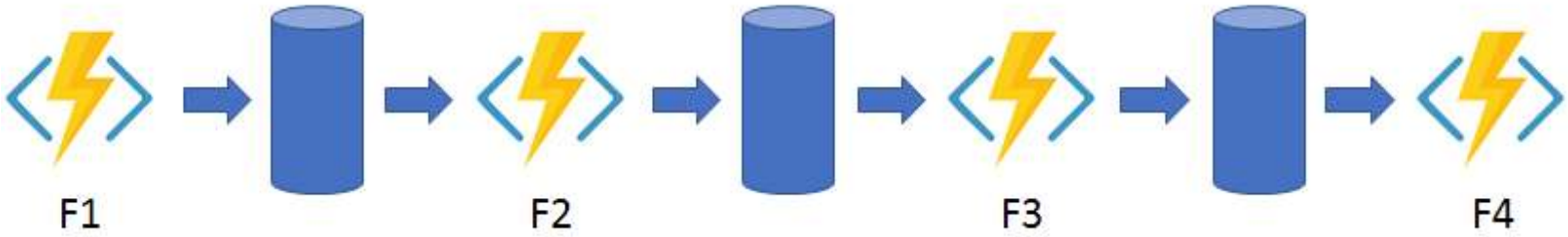
DEMO

How to create a function app locally

Durable Functions extension

- The primary use case for Durable Functions is simplifying complex, stateful coordination problems in serverless applications

Durable Functions: function chaining

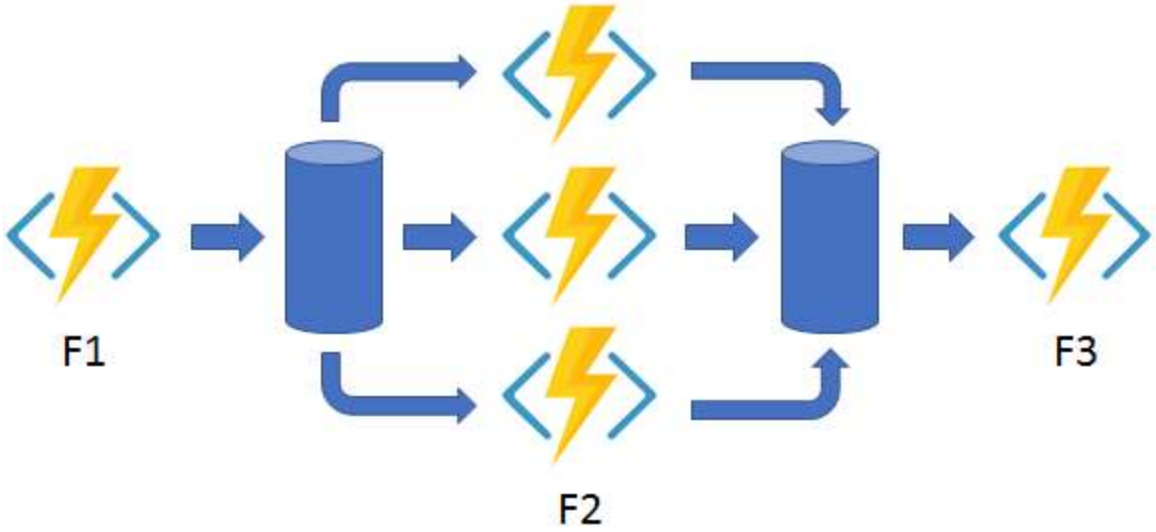


Durable Functions: function chaining

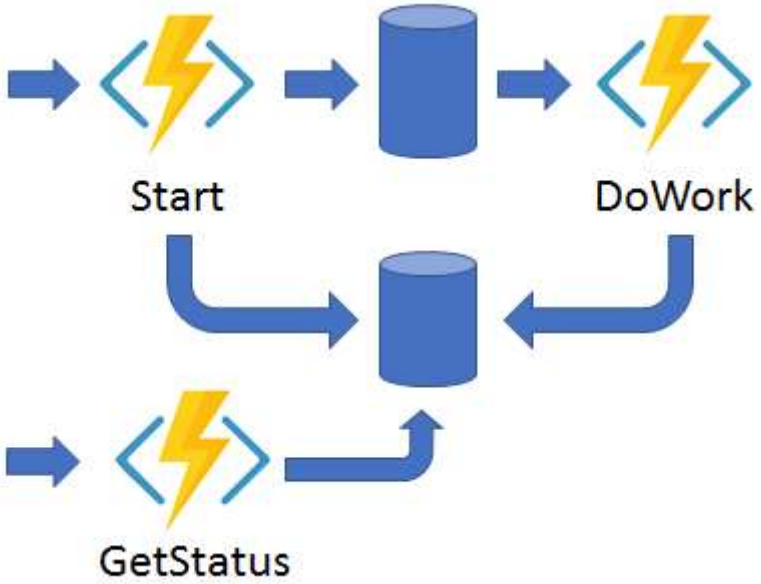
```
const df = require("durable-functions");


module.exports = df(function*(ctx) {
          const x = yield ctx.df.callActivityAsync("F1");
          const y = yield ctx.df.callActivityAsync("F2", x);
          const z = yield ctx.df.callActivityAsync("F3", y);
          return yield ctx.df.callActivityAsync("F4", z);
});
```

Durable Functions: fan out / fan in



Durable Functions: async API



DEMO

Other features

- Proxy
- Slot
- Monitoring