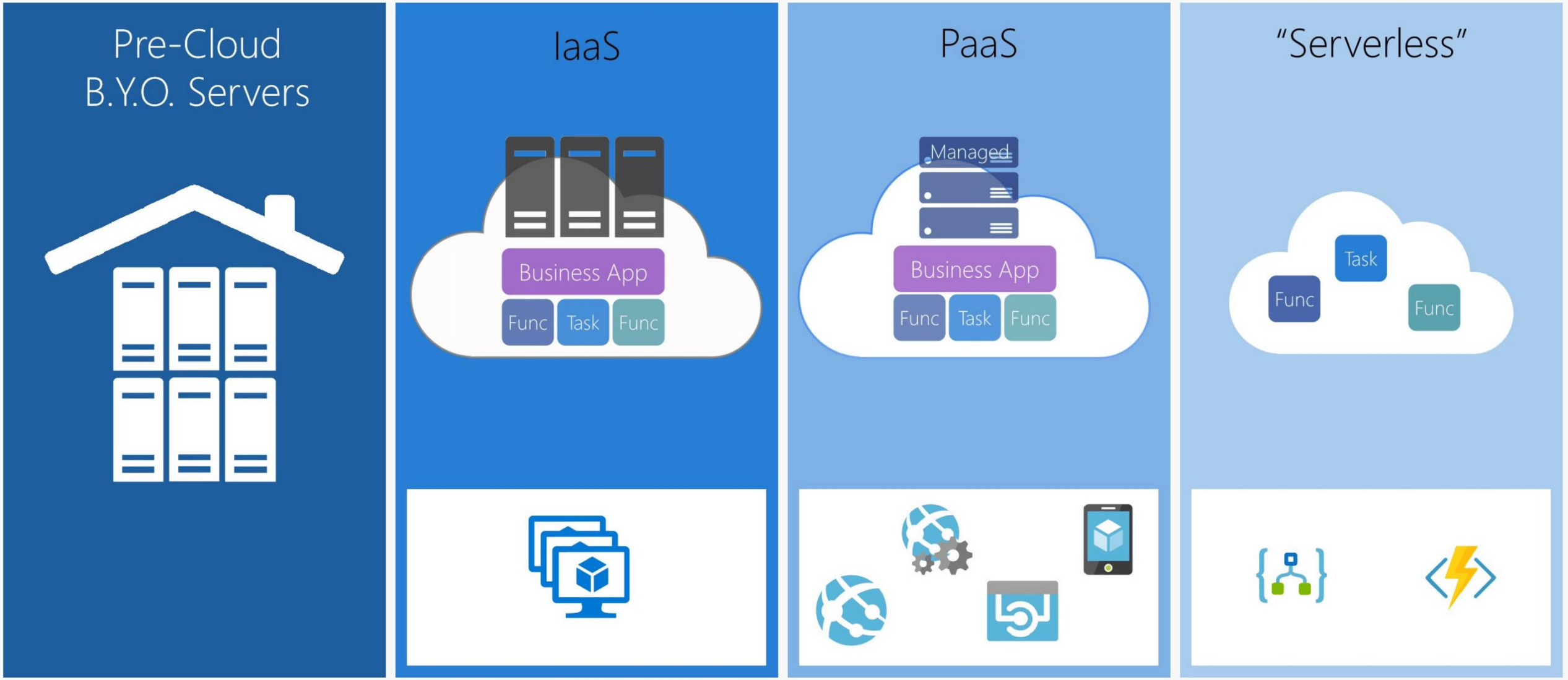
.NET Core
a Azure Functions v2

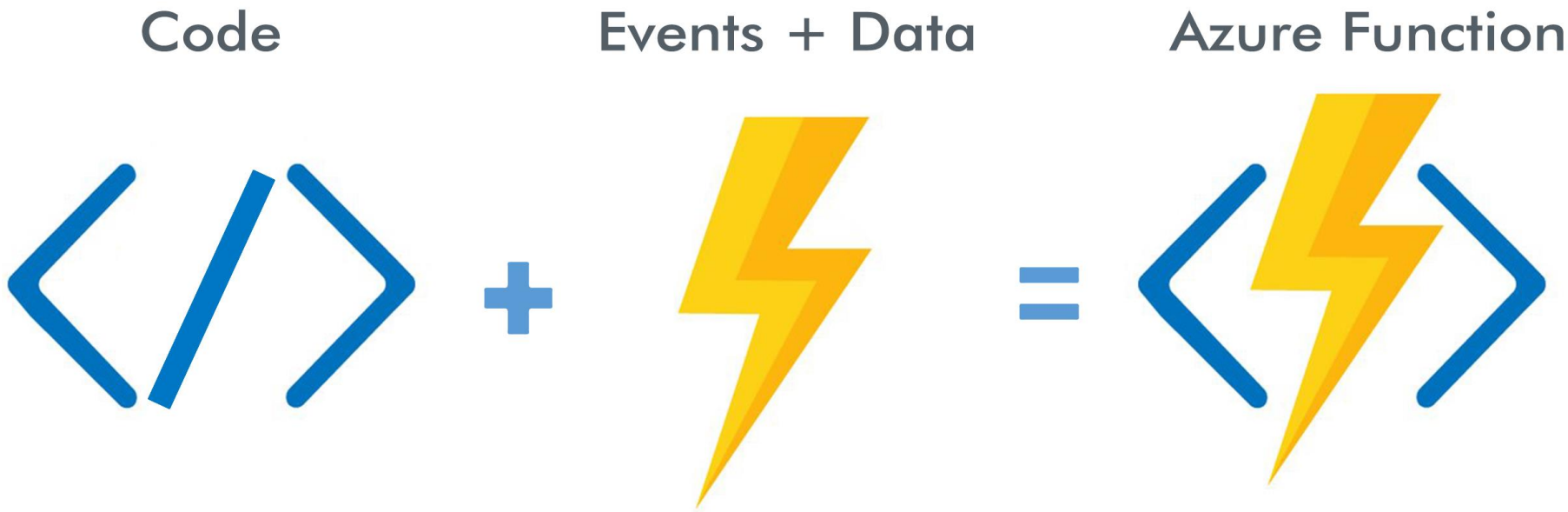
VÁCLAV JIROVSKÝ, DOTNET TALKS 17.1.2019 – VERZE PRO INTERNET



Pokud chci provozovat svoji aplikaci v on-premise, musím řešit provozování celého serveru, internetového připojení, zálohy, nastavení OS apod. V případě IaaS (VM) řeším pak jenom od úrovně OS výše.
U PaaS (např. Azure Web App) řeším už pouze svoji aplikaci – routování, připojení.
Serverless = nádstavba nad PaaS, kdy pouze předám svůj kód (jednotlivé funkce) a hostitel zařídí jeho vykonání.

Co jsou Azure Functions

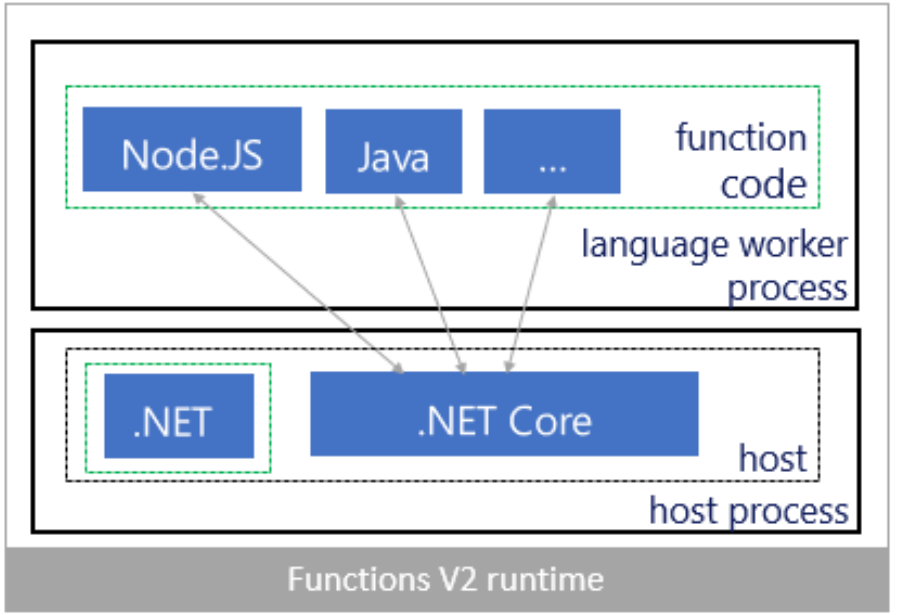
„Execute your code based on events you specify“



Serverless technologie od MS se nazývá Azure Functions. Hostiteli předáte váš zdrojový kód, podmínky kdy se má vykonat a s čím potřebuje komunikovat.

Podporované technologie (ve v2)

- C#, F# (.NET Core 2.0)
- JavaScript + TypeScript (Node 8, 10)
- Java (Java 8)
- Python (Python 3.6)



Kód funkce pro Azure Functions můžete psát v těchto jazycích. Hostitel je podle vybraného jazyku na OS Windows nebo Linux.

Co potřebuji pro vývoj

- Windows/Mac/Linux
- Visual Studio, Visual Studio Code,…
- Azure Functions Core Tools – *func*
  - Npm
- Azure Storage Emulator / Azure Storage account

Pro vývoj Azure Functions potřebujete hlavně SDK - Azure Functions Core Tools – func(.exe), které řeší vše od vytváření funkcí (pomocí šablon) až po debuggování. Dále potřebujete Azure Storage (např. pro použití Timer triggeru).

Triggers

- Event-driven architecture

- Událost, která vyvolá spuštění funkce

  - Např.: HTTP zavolání, nový záznam ve frontě, CRON,…

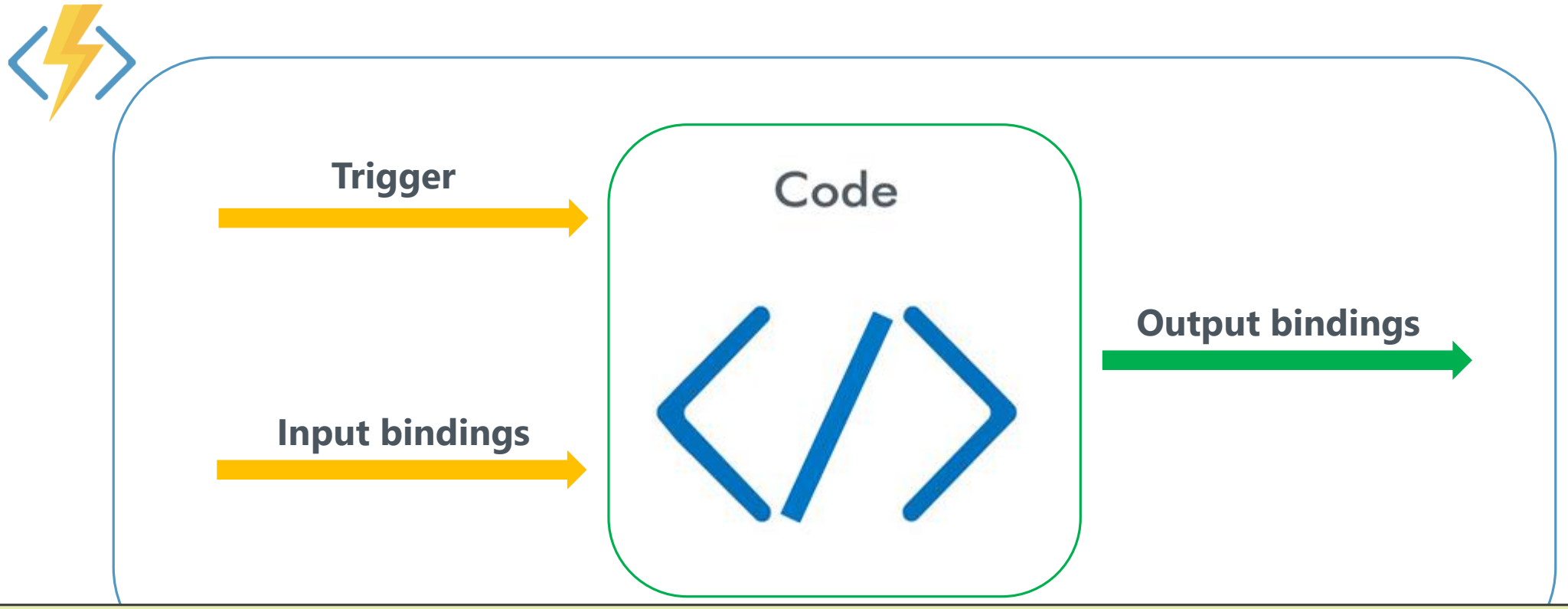
Funkce je vypnutá, dokud není zavolaná nějakou externí událostí. Tato událost se nazývá „trigger“.

Bindings

- Hotové datové konektory na další služby pro použití ve funkci
- Input vs. output
- Ve stylu Dependency Injection

Aby funkce mohla komunikovat s okolním světem, vytvořil Microsoft datové konektory, které mohu použít (po předání connection stringu/API klíče).
Input binding – datový zdroj, ze kterého potřebuje funkce čerpat data pro svůj běh (např. CosmosDB tabulka)
Output binding – datový zdroj, kam funkce dává nějaký svůj výstup (např. ServiceBus Queue)

Bindings & Trigger



Aby Azure Functions mohla fungovat, potřebuje mít definovaný trigger a zdrojový kód. Input a output bindings jsou nepovinné. Funkce musí mít pouze jeden trigger.

Bindings v Azure Functions v2

- Blog Storage I, O
- Cosmos DB I, O
- HTTP (webhook) O
- Graph Excel tables I, O
- Graph OneDrive files I, O
- Graph Outlook email O
- Graph Events I, O
- Microsoft Graph Auth tokens I
- Queue storage O
- SendGrid O
- Service Bus O
- Table storage O
- Twilio O

I = Input binding, O = Output binding

V Azure Functions v2 jsou dostupné tyto datové zdroje.
Příklad použití: Funkce bude kopírovat obsah kolekce z Cosmos DB (input binding) do blob storage (output binding) a pošle administrátorovi emailem report přes SendGrid (output binding).

Triggers v Azure Functions v2

- Blog Storage
- Cosmos DB
- Event Grid
- Event Hubs
- HTTP (Webhook)
- Microsoft Graph Events
- Queue storage
- Service Bus
- Timer

V Azure Functions v2 jsou dostupné tyto události, které mohou spustit funkci.
Příklad použití z předchozí slide můžeme nastavit, aby se spouštěl každou hodinu (Trigger Timer).

DEMO – vytvoření nové funkce

Visual studio Code ->template

V otevřeném Azure Functions projektu ve Visual Studio Code kliknete na Azure extension a zde je tlačítko „New Function“. Wizard vás provede vybráním triggeru a pojmenováním funkce.

DEMO – jednoduchá HTTP trigger funkce

https://github.com/vjirovsky/azure-functions-v2-demo/tree/master/01-basic-usage

DEMO – output data binding

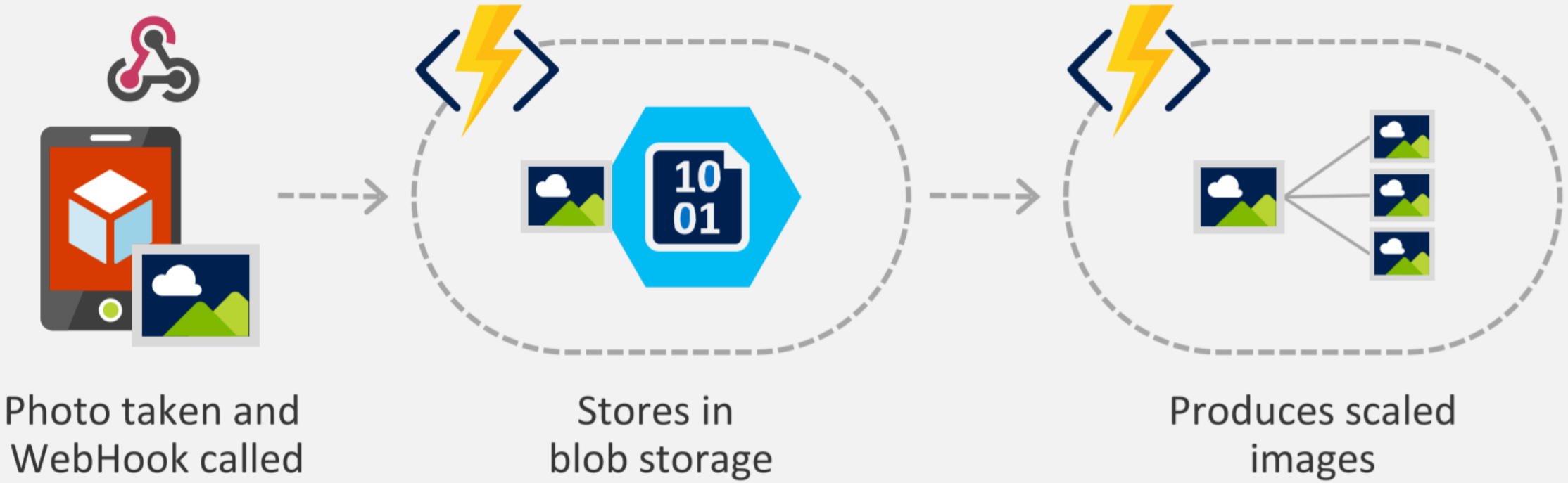
https://github.com/vjirovsky/azure-functions-v2-demo/tree/master/02-bindings-demo/output-demo

Service Bus Output binding + přístupové klíče

DEMO – Service Bus trigger

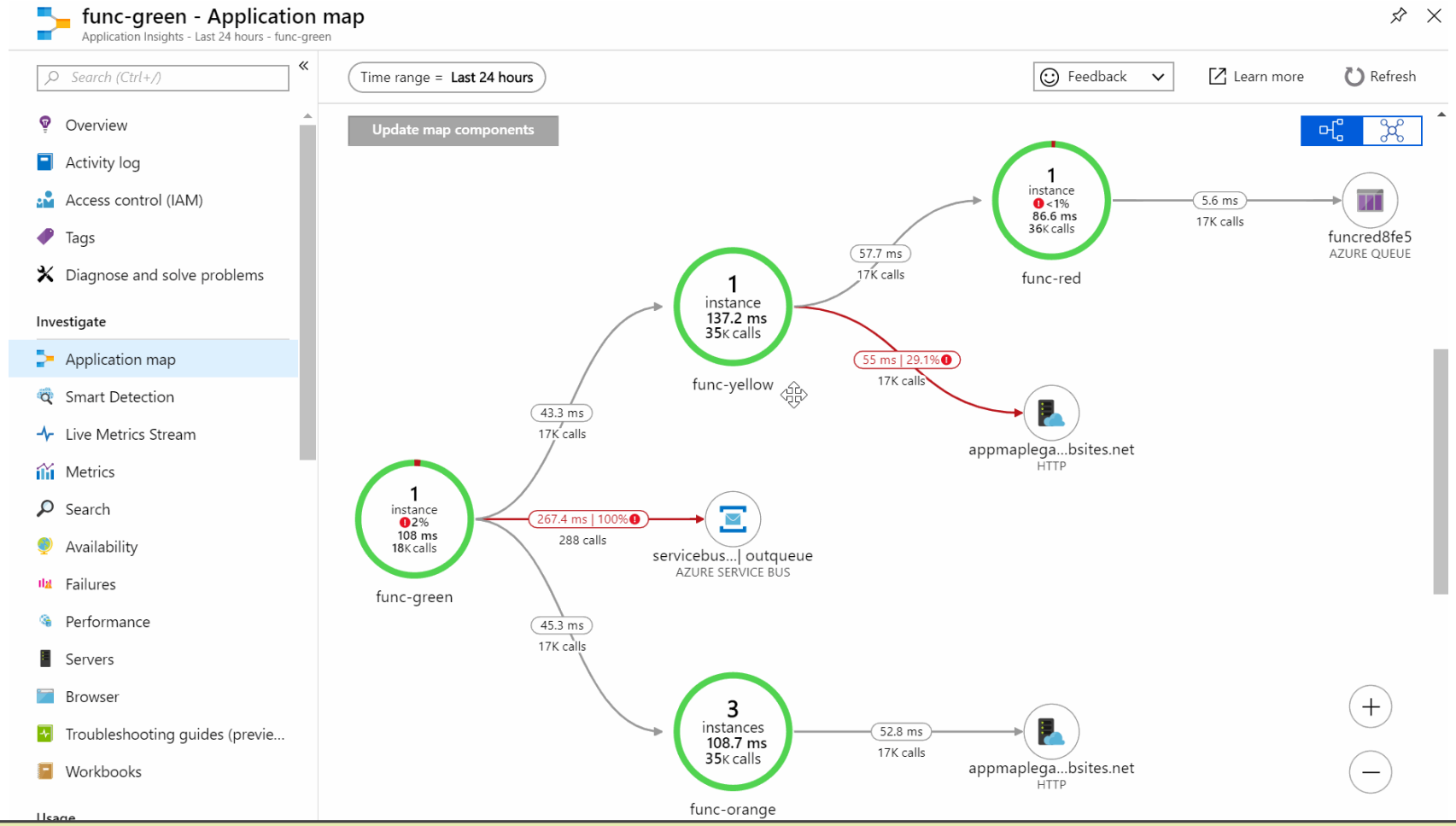
https://github.com/vjirovsky/azure-functions-v2-demo/tree/master/02-bindings-demo/trigger-demo

Jedním z příkladu použití je vytvoření tasku, který pročistí každých 15 minut tabulku v databázi.



Dalším příkladem použití může být zřetězení více funkcí za sebou. Na příkladu dojde po uploadu nového obrázku přes REST API nahráním do blob storage první funkcí a druhá funkce na základě nového obrázku v blob storage připraví jeho verze pro různé displeje.

Monitoring – App Insights



V Azure Functions jsou plně integrované Azure App Insights, takže pro monitoring a debugging můžete použít jejich funkce. Včetně zobrazení provázanosti funkcí (i na externí služby).

Rozšíření Durable Functions

- Standardně jsou Azure Functions bezestavové
- Řeší stav, checkpointy, restarty jednotlivých jobů
- https://docs.microsoft.com/en-us/azure/azure-functions/durable/durable-functions-create-first-csharp

Rozšíření Durable Functions přináší stavové funkce do bezestavového prostředí. Toto rozšíření za vás řeší stavy jednotlivých tasků, checkpointy a případné restarty v případě problémů při zpracování jobu.

Kde provozovat Azure Functions

- Serverless v Azure
- V Docker kontejneru
  - V cloudu (v Azure nebo např. AWS Fargate)
  - On premise

Azure Functions můžete provozovat v serverless prostředí, kdy se nemusíte o hostitele vůbec starat. Další možností je v Docker kontejneru, který pak můžete provozovat v cloudu nebo i on-premise.

DEMO

Azure serverless deployment, management

DEMO

Vytvoření + popis Dockerfile, spuštění v Dockeru

https://github.com/vjirovsky/azure-functions-v2-demo/tree/master/03-docker/original

https://github.com/vjirovsky/azure-functions-v2-demo/tree/master/03-docker/vaseks-host

Provozování on-premise

- Nutné Azure Storage připojení:
  - Azure Storage – být online
  - Azure Storage Emulator – nespolehlivý, padá
  - Azurite – open-source emulátor Azure Storage, Queue, Table

Azure Functions potřebují i pro on-premise provoz připojení k Azure Storage, kam si ukládají metadata (v případě použití Timer kdy zavolat funkci příště, zámky apod.). Možností je více - připojit online Azure Storage, použít Azure Storage Emulator a nebo Azurite, open-source emulátor i pro Linux.

Pricing v Azure serverless

- Consuption plan
  - Platíte za každé zavolání a přenos dat
  - Azure automaticky škáluje

- App Service plan
  - Stejné jako Web Apps
  - Dedikované prostředky

+ Azure Storage

Pro pricing Azure Functions můžete použít buď microbilling za konzumaci nebo App Service plán. Výběr záleží na smyslu mojí aplikace.

Výhody serverless

- Řešíte aplikaci, ne servery a prostředí
- Snížení času DevOps
- Rychlejší vývoj a nasazení
- Snadné škálování

Rozdíly oproti Azure Functions v1

- Prozatím chybí podpora pro PHP, Bash, cmd, PowerShell
- Všechny funkce v jedné app musí být ve stejném jazyce
- Kromě *HTTP* a *Timer* se musí binding zaregistrovat

Další informace

- https://docs.microsoft.com/en-us/azure/azure-functions/
- https://blog.vjirovsky.cz/running-azure-functions-on-your-infrastructure/
- stackoverflow *#azure-functions*

Díky za pozornost

@vjirovsky