Lab 6 - Azure Functions

Model Chmurowy i Serverless

Materiały na kolokwium - Część 1

7 grudnia 2025

Spis treści

1 Model Chmurowy Azure

1.1 Typy Usług Chmurowych

IaaS - Infrastructure as a Service

Definicja: Wynajmujesz infrastrukturę (VM, sieci, storage)
Charakterystyka:

- Pełna kontrola nad systemem operacyjnym
- Zarządzasz aktualizacjami i konfiguracją
- Odpowiedzialność za security patches
- Elastyczne skalowanie zasobów
- Maksymalna elastyczność i kontrola

Przykłady w Azure:

- Azure Virtual Machines
- Azure Virtual Network
- Azure Storage (raw)
- Azure Disk Storage

Kiedy używać:

- Migracja legacy aplikacji (lift-and-shift)
- Potrzebujesz pełnej kontroli nad OS
- Specyficzne wymagania konfiguracyjne
- Aplikacje wymagające custom software

Odpowiedzialność:

- TY: OS, middleware, runtime, aplikacje, dane
- AZURE: hardware, networking, datacenter

PaaS - Platform as a Service

Definicja: Platforma do deploymentu aplikacji bez zarządzania infrastrukturą
Charakterystyka:

- Azure zarządza infrastrukturą
- Automatyczne aktualizacje i patche
- Built-in high availability
- Focus na kod aplikacji
- Szybsze time-to-market

Przykłady w Azure:

- Azure App Service
- Azure Functions
- Azure SQL Database
- Azure Cosmos DB

Kiedy używać:

- Standardowe web aplikacje
- Chcesz szybko deployować
- Nie chcesz zarządzać infrastrukturą
- Modern cloud-native apps

Odpowiedzialność:

- TY: aplikacje, dane
- AZURE: wszystko inne (OS, runtime, networking, storage)

SaaS - Software as a Service

Definicja: Gotowe aplikacje dostępne przez internet
Charakterystyka:

- Kompletne rozwiązanie software
- Dostęp przez przeglądarkę
- Zero maintenance
- Subscription model

Przykłady:

- Microsoft 365 (Word, Excel, Outlook)
- Dynamics 365
- Azure DevOps
- GitHub

Odpowiedzialność:

- TY: dane i dostęp użytkowników
- AZURE: wszystko inne

1.2 Serverless Computing

Ważne!

Serverless NIE oznacza "bez serwera"!
Serwery istnieją, ale:

- Nie zarządzasz nimi
- Automatyczne skalowanie (od 0 do nieskończoności)
- Płacisz tylko za rzeczywiste użycie
- Event-driven execution
- Abstrakcja infrastruktury

Charakterystyka Serverless

Event-driven:

- Kod uruchamia się w odpowiedzi na zdarzenia
- HTTP request, timer, message w kolejce, nowy plik

Stateless:

- Każde wykonanie jest niezależne
- Stan trzeba przechowywać zewnętrznie (database, storage)
- Brak shared memory między wykonaniami

Ephemeral:

- Krótkotrwałe wykonania
- Instancje powstają i giną dynamicznie

Zalety Serverless

- **Zero server management** - Azure zarządza wszystkim
- **Automatic scaling** - od 0 do tysięcy instancji automatycznie
- **Pay-per-execution** - nie płacisz za idle time
- **Built-in availability** - automatyczna redundancja i failover
- **Faster time to market** - focus na logikę biznesową
- **Reduced operational costs** - brak kosztów utrzymania infrastruktury
- **Infinite scalability** - teoretycznie unlimited

Wady Serverless

- **Cold start** - pierwsze uruchomienie trwa dłużej (kilka sekund)
- **Execution time limits** - max 10 min (Consumption Plan)
- **Stateless** - trzeba przechowywać stan zewnętrznie
- **Vendor lock-in** - trudniejsza migracja między cloud providerami
- **Debugging complexity** - trudniejsze debugowanie distributed system
- **Testing** - trudniejsze testowanie lokalnie
- **Monitoring** - wymaga dobrych narzędzi do observability

Cold Start Problem

Co to jest Cold Start?

- Opóźnienie przy pierwszym uruchomieniu funkcji
- Azure musi: przydzielić kontener, załadować runtime, załadować kod
- Czas: zwykle 1-5 sekund, może być dłużej dla dużych aplikacji

Kiedy występuje:

- Pierwsza próba wywołania funkcji
- Funkcja nie była używana przez kilka minut (idle)
- Scaling out - nowe instancje potrzebują warm-up

Jak zminimalizować:

- Premium Plan - pre-warmed instances (zawsze gotowe)
- Zmniejsz rozmiar deployment package
- Użyj języka z szybkim startem (C# compiled, nie Python)
- Warming techniques (keep-alive requests)

2 Azure Functions

2.1 Podstawowe Koncepcje

Czym są Azure Functions?

Azure Functions to serverless compute service do uruchamiania event-driven kodu.
Definicja komponentów:

- **Function** - jednostka kodu wykonująca konkretne zadanie
- **Trigger** - zdarzenie uruchamiające funkcję (tylko 1 trigger na funkcję)
- **Binding** - deklaratywne połączenie z zasobami (input/output)
- **Function App** - kontener dla wielu funkcji (shared context)

Function App jako jednostka:

- Grupuje powiązane funkcje
- Shared configuration (app settings)
- Shared resources (memory, CPU)
- Deploy razem
- Scale razem

Supported Languages

GA (Generally Available):

- C# (. NET 6, .NET 7, . NET 8)
- JavaScript/TypeScript (Node.js)
- Python (3.7, 3.8, 3. 9, 3.10, 3.11)
- Java (8, 11, 17)
- PowerShell (7. 2)

Preview:

- Custom handlers (any language)

2.2 Plany Hostingowe

Consumption Plan (Serverless)

Charakterystyka:

- Dynamiczne skalowanie (automatyczne od 0)
- Płatność za: liczbę wykonań + czas wykonania + pamięć użyta
- 1 milion wykonań FREE miesięcznie
- 400,000 GB-seconds FREE miesięcznie
- Max czas wykonania: 10 minut (default 5 min, configurable)
- Max instancji: 200 (domyślnie)
- Cold start występuje

Pricing model:

- $0.20 per milion executions
- $0.000016 per GB-s

Kiedy używać:

- Nieregularne/nieprzewidywalne obciążenie
- Krótkie, szybkie operacje ($<$ 5 min idealnie)
- Optymalizacja kosztów (pay only for what you use)
- Proof of concept / development

Premium Plan (Elastic Premium)

Charakterystyka:

- Pre-warmed instances - zawsze gotowe (NO cold start)
- Unlimited execution time (no timeout)
- VNET connectivity (private networking)
- Większa moc obliczeniowa (premium hardware)
- Faster scaling
- Płatność za: uptime instancji (vCore-seconds)

Instance sizes:

- EP1: 1 vCore, 3. 5 GB RAM
- EP2: 2 vCore, 7 GB RAM
- EP3: 4 vCore, 14 GB RAM

Kiedy używać:

- Aplikacje wymagające stałej wydajności
- Potrzebujesz VNET integration (private endpoints)
- Długie operacje ($>$ 10 min)
- Wysokie wymagania wydajnościowe
- Production workloads gdzie cold start niedopuszczalny

Dedicated (App Service) Plan

Charakterystyka:

- Funkcje działają na dedykowanych VM App Service
- Przewidywalne koszty (stała miesięczna opłata)
- Shared z innymi App Services (można współdzielić plan)
- Pełna kontrola nad skalowaniem (manual lub auto-scale rules)
- Brak limitów wykonania
- Brak cold start (jeśli always on enabled)

Kiedy używać:

- Już masz App Service Plan (optimization)
- Stałe, przewidywalne obciążenie 24/7
- Potrzebujesz pełnej kontroli
- Existing underutilized VM
- Long-running functions

Always On:

- Ustawienie App Service Plan
- Funkcje pozostają załadowane w pamięci
- Eliminuje cold start
- WYMAGANE dla non-HTTP triggers na Dedicated Plan

Ważne!

Porównanie planów - koszty:

- Consumption: najtańszy dla sporadycznego użycia
- Premium: przewidywalne koszty, no cold start
- Dedicated: najlepszy gdy już masz App Service

Cold start:

- Consumption: TAK (zawsze)
- Premium: NIE (pre-warmed)
- Dedicated: NIE (z Always On)

3 Triggersy Azure Functions

3.1 HTTP Trigger

HTTP Trigger - Podstawy

Definicja: Funkcja wywoływana przez żądanie HTTP/HTTPS
Zastosowanie:

- RESTful API endpoints
- Webhooks (GitHub, Stripe, external services)
- Integracje z zewnętrznymi systemami
- Serverless web applications
- Microservices architecture

Supported HTTP Methods:

- GET - pobieranie danych
- POST - tworzenie zasobów
- PUT - aktualizacja zasobów (full update)
- PATCH - częściowa aktualizacja
- DELETE - usuwanie zasobów
- HEAD, OPTIONS - metadata

Routing:

- Default: `https://{app}.azurewebsites.net/api/{function}`
- Custom route: można zmienić `/api/` prefix
- Route parameters: `/users/{id}`

Authorization Levels

Anonymous:

- Brak autentykacji
- Publiczny dostęp
- Każdy może wywołać funkcję
- Use case: publiczne API, webhooks bez auth

Function:

- Wymaga function-specific key
- Key w query string: `?code=<KEY>`
- Lub w header: `x-functions-key: <KEY>`
- Każda funkcja ma swój unikalny klucz
- Use case: standard API security

Admin:

- Wymaga master key (host key)
- Dostęp do WSZYSTKICH funkcji w Function App
- Najwyższy poziom uprawnień
- Use case: administrative operations

Ważne!

Best practices:

- NIE używaj Anonymous dla production API (chyba że public)
- Function keys powinny być rotowane regularnie
- Przechowuj keys w Key Vault
- Rozważ Azure AD authentication dla enterprise apps

HTTP Request i Response

Request components:

- Method (GET, POST, etc.)
- Headers (Content-Type, Authorization, etc.)
- Query parameters (? param=value)
- Route parameters (/users/123)
- Body (JSON, XML, form data)

Response:

- Status code (200, 404, 500, etc.)
- Headers
- Body (JSON najpopularniejsze)

Content negotiation:

- JSON - `Content-Type: application/json`
- XML - `Content-Type: application/xml`
- Plain text - `Content-Type: text/plain`

3.2 Timer Trigger

Timer Trigger - Podstawy

Definicja: Funkcja uruchamiana według harmonogramu (scheduled execution)
Zastosowanie:

- Scheduled jobs (cron jobs replacement)
- Okresowe przetwarzanie danych (batches)
- Cleanup tasks (usuwanie starych danych)
- Generowanie raportów (daily, weekly)
- Monitoring i health checks
- Backup operations
- Data synchronization
- Sending periodic notifications

Charakterystyka:

- Singleton execution - tylko jedna instancja w danym momencie
- UTC timezone (default, można zmienić)
- Może się opóźnić przy cold start (pierwsze uruchomienie)
- Nie wymaga żadnych external dependencies

CRON Expression

Format Azure Functions CRON:

```
{second} {minute} {hour} {day} {month} {day-of-week}
```

6 pól (w Azure), nie 5 jak w Linux!
Wildcards i operatory:

- `*` - any value (każda wartość)
- `,` - value list separator (0,30 = 0 i 30)
- `-` - range (1-5 = od 1 do 5)
- `/` - increment (*/5 = co 5)

Day of week:

- 0 lub 7 = Sunday
- 1 = Monday
- 6 = Saturday
- Można też: MON, TUE, WED, THU, FRI, SAT, SUN

CRON Expression - Przykłady

Co 5 minut:

```
0 */5 * * * *
```

Co godzinę (na początku godziny):

```
0 0 * * * *
```

Codziennie o 9:00:

```
0 0 9 * * *
```

W dni robocze o 9:00:

```
0 0 9 * * MON-FRI
```

Pierwszego dnia miesiąca o północy:

```
0 0 0 1 * *
```

Co 30 sekund:

```
*/30 * * * * *
```

Każdego poniedziałku i piątku o 14:30:

```
0 30 14 * * MON,FRI
```

Co 6 godzin:

```
0 0 */6 * * *
```

Ważne!

Timezone:

- Default: UTC
- Zmiana timezone: app setting `WEBSITE_TIME_ZONE`
- Wartości: "Central European Standard Time", "US Eastern", etc.
- Lista: Microsoft time zone index values

Uwagi:

- Timer trigger może się opóźnić przy scale-down
- W Consumption Plan możliwe opóźnienia przy cold start
- Dla critical timing użyj Premium Plan

3.3 Blob Trigger

Blob Trigger - Podstawy

Definicja: Funkcja uruchamiana gdy blob jest dodany lub zaktualizowany
Zastosowanie:

- Przetwarzanie uploadowanych plików
- Generowanie thumbnails obrazów
- Analiza dokumentów (PDF, Word parsing)
- ETL pipelines (Extract, Transform, Load)
- Video/audio transcoding
- Virus scanning nowych plików
- Image recognition (AI/ML)
- File format conversion

Triggering events:

- New blob created
- Existing blob updated (overwritten)
- NIE triggeruje na: delete, metadata changes

Blob Trigger - Monitoring Mechanism

Jak działa detection:

1. Azure Functions skanuje logi Blob Storage
2. Wykrywa zmiany w kontenerze
3. Uruchamia funkcję dla nowych/zmodyfikowanych blobów

Latency (opóźnienie):

- Consumption Plan: może być opóźnienie do kilku minut
- Szczególnie dla rzadko używanych Function Apps
- Cold start dodatkowo wydłuża czas reakcji

Premium/Dedicated Plan:

- Może używać Event Grid (szybsze)
- Real-time lub near-real-time detection
- Lepsza wydajność dla high-volume scenarios

Ważne!

Dla real-time processing:
Zamiast Blob Trigger użyj **Event Grid Trigger**!
Event Grid advantages:

- Near-instant reaction (milisekundy)
- Brak polling overhead
- Lepsze dla high-throughput scenarios
- Events dla: create, delete, rename

Path Pattern

Blob path format:

```
container-name/{blob-name}
```

Przykłady:
Wszystkie blobs w kontenerze:

```
samples-workitems/{name}
```

W określonym folderze:

```
samples-workitems/uploads/{name}
```

Tylko pliki JPG:

```
images/{name}.jpg
```

Binding expression - extract metadata:

```
images/{date}/{category}/{filename}. jpg
```

Możesz potem użyć: date, category, filename jako parametry!

Blob Metadata

Informacje dostępne w funkcji:

- Blob name (nazwa pliku)
- Blob URI (pełna ścieżka)
- Content type (MIME type)
- Size (rozmiar w bajtach)
- Last modified timestamp
- ETag (version identifier)

Możesz też:

- Czytać content blobu (binary data)
- Czytać metadata (custom properties)
- Stream duże pliki (nie ładuj wszystkiego do pamięci)

3.4 Queue Trigger

Queue Trigger - Podstawy

Definicja: Funkcja uruchamiana gdy pojawi się wiadomość w Azure Queue Storage
Zastosowanie:

- Asynchroniczne przetwarzanie zadań
- Load leveling - wyrównywanie obciążenia (buffering)
- Decoupling komponentów aplikacji
- Background processing
- Order processing workflows
- Email/notification sending
- Batch data processing
- Integration patterns

Architektura:

1. Producer → dodaje message do queue
2. Queue → przechowuje message
3. Consumer (Function) → przetwarza message
4. Delete → usunięcie po sukcesie

Queue Trigger - Polling Mechanism

Jak działa:

- Azure Functions automatycznie polluje kolejkę
- Adaptive polling algorithm:
  - Szybkie polling gdy są wiadomości
  - Wolniejsze gdy kolejka pusta (save costs)
- Batch dequeue - może pobrać wiele wiadomości naraz
- Parallel processing - wiele instancji funkcji równolegle

Scaling:

- Azure automatycznie skaluje liczbę instancji
- Bazuje na queue depth (ile wiadomości w kolejce)
- Więcej messages → więcej instances
- Max instancji zależy od planu (Consumption: 200)

Visibility Timeout

Co to jest Visibility Timeout?

- Gdy message jest dequeued → staje się invisible dla innych
- Default: 30 sekund
- Daje czas na przetworzenie message
- Inne consumery nie widzą tej wiadomości

Scenariusze:
Success:

1. Dequeue message (invisible 30s)
2. Process message (np. 10s)
3. Delete message przed timeout
4. Success - message gone

Failure:

1. Dequeue message (invisible 30s)
2. Process crashes (exception thrown)
3. Timeout expires (30s)
4. Message wraca do queue
5. Retry (dequeue count++)

Poison Messages

Co to jest Poison Message?

- Wiadomość która powoduje błąd podczas przetwarzania
- Repeatedly fails (wielokrotne błędy)
- Może zablokować kolejkę

Azure Automatic Handling:

1. Każdy dequeue zwiększa `DequeueCount`
2. Po 5 nieudanych próbach (default)
3. Azure automatycznie przenosi do poison queue
4. Poison queue name: `{originalqueue}-poison`

Poison queue handling:

- Wymaga ręcznej obsługi (monitoring)
- Można napisać osobną funkcję do analizy
- Investigate: dlaczego message failed?
- Fix: napraw dane lub kod
- Requeue lub discard

Configurable:

- Można zmienić max retry count (default 5)
- W host.json: `maxDequeueCount`

Ważne!

Best practices dla Queue Trigger:

- Idempotency - funkcja powinna być idempotentna (można uruchomić wielokrotnie bezpiecznie)
- Message size - max 64 KB (dla większych użyj Service Bus)
- TTL - ustaw sensowny Time-To-Live
- Monitoring - monitoruj poison queue
- Error handling - proper try-catch i logging

4 Bindingi

4.1 Podstawy Bindingów

Co to są Bindings?

Definicja: Deklaratywny sposób połączenia funkcji z zasobami Azure
Zalety:

- Brak kodu connection management
- Mniej boilerplate code
- Automatic retry logic
- Connection pooling (optymalizacja)
- Credentials z App Settings (security)
- Separation of concerns

Typy:

- **Input Binding** - czytanie danych
- **Output Binding** - zapisywanie danych
- **Trigger** - specjalny typ input binding (uruchamia funkcję)

Konfiguracja:

- function.json (JavaScript, Python)
- Attributes (C#)
- Annotations (Java)

4.2 Input Bindings

Input Bindings - Charakterystyka

Definicja: Pobieranie danych z external sources do funkcji
Dostępne źródła:

- Blob Storage - odczyt pliku
- Queue Storage - peek message (bez usuwania)
- Table Storage - pobranie entity/entities
- Cosmos DB - query dokumentów
- SQL Database - SELECT query
- SignalR - connection info

Przykład użycia:

- HTTP Trigger otrzymuje ID
- Input Binding pobiera dane z Cosmos DB dla tego ID
- Funkcja przetwarza i zwraca

4.3 Output Bindings

Output Bindings - Charakterystyka

Definicja: Zapisywanie danych do external destinations z funkcji
Dostępne destinations:

- Blob Storage - zapis pliku
- Queue Storage - wysłanie wiadomości
- Table Storage - insert/update entity
- Cosmos DB - create/update document
- Event Hub - publish events
- Event Grid - publish events
- SignalR - broadcast messages
- SendGrid - wysyłanie emaili
- Twilio - wysyłanie SMS

Multiple output bindings:

- Jedna funkcja może mieć wiele output bindings
- Przykład: zapisz do Blob + wyślij do Queue + update Table

Ważne!

Trigger vs Input Binding:
Trigger:

- Uruchamia funkcję
- Tylko 1 trigger na funkcję
- Event-driven

Input Binding:

- Dostarcza dane do funkcji
- Wiele input bindings możliwe
- Data retrieval