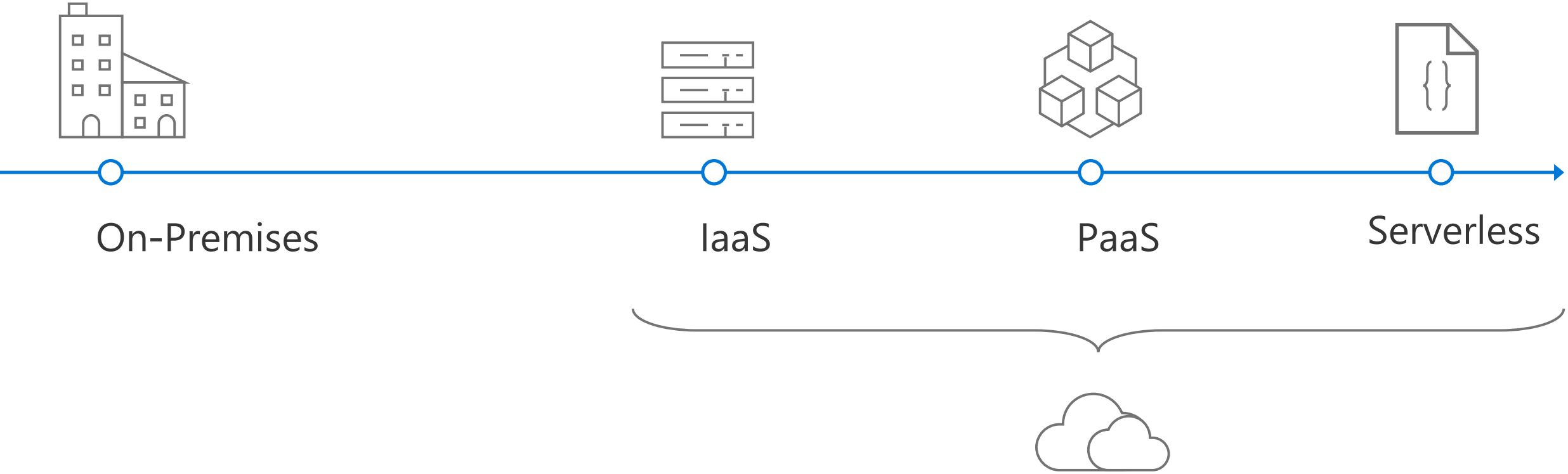
클라우드 서버리스(Serverless) 개발 실습

김태영

CSE,
한국마이크로소프트

Microsoft

응용프로그램 플랫폼의 진화



백업을 유지하기 위해서 어떤 장치를 사용해야 하나

우리 비즈니스에 필요한 서버의 사양은 정확히 어느 정도인가

새로운 코드를 서버에 어떻게 배포할 수 있는가

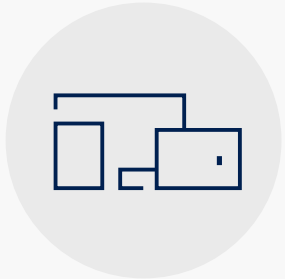
서버의 하드웨어에서 장애가 나면 어떻게 해야 하는가

어떤 패키지들이 서버에 설치되어야 하는가

우리의 앱은 누가 관리 및 모니터링 하는가

얼마나 자주 서버를 백업해야 하는가

어느 정도 규모의 서버를 구입해야 하는가



어떻게 우리의 앱을 스케일 할 수 있을까?

어떻게 하면 서버의 활용도를 높일 수 있을까

2차 네트워크 연결이 필요할까

서버에 물리적으로 접근이 가능한 사람은 누구인가

어떤 OS를 사용해야 하는가

우리 서버는 안전한 곳에 놓여있는가

누가 서버를 모니터링하지?

UPS가 필요할까?

전원 공급이 끊어지면 어떻게 대응해야 하나

어떤 스토리지를 사용해야 하는가

얼마나 많은 서버가 필요할까?

새로운 서버를 프로비전하는 데 얼마나 오래 걸리는가?

얼마나 자주 패치를 수행해야만 하는가

앱을 어떻게 동적으로 구성할 수 있는가?

On-Premises

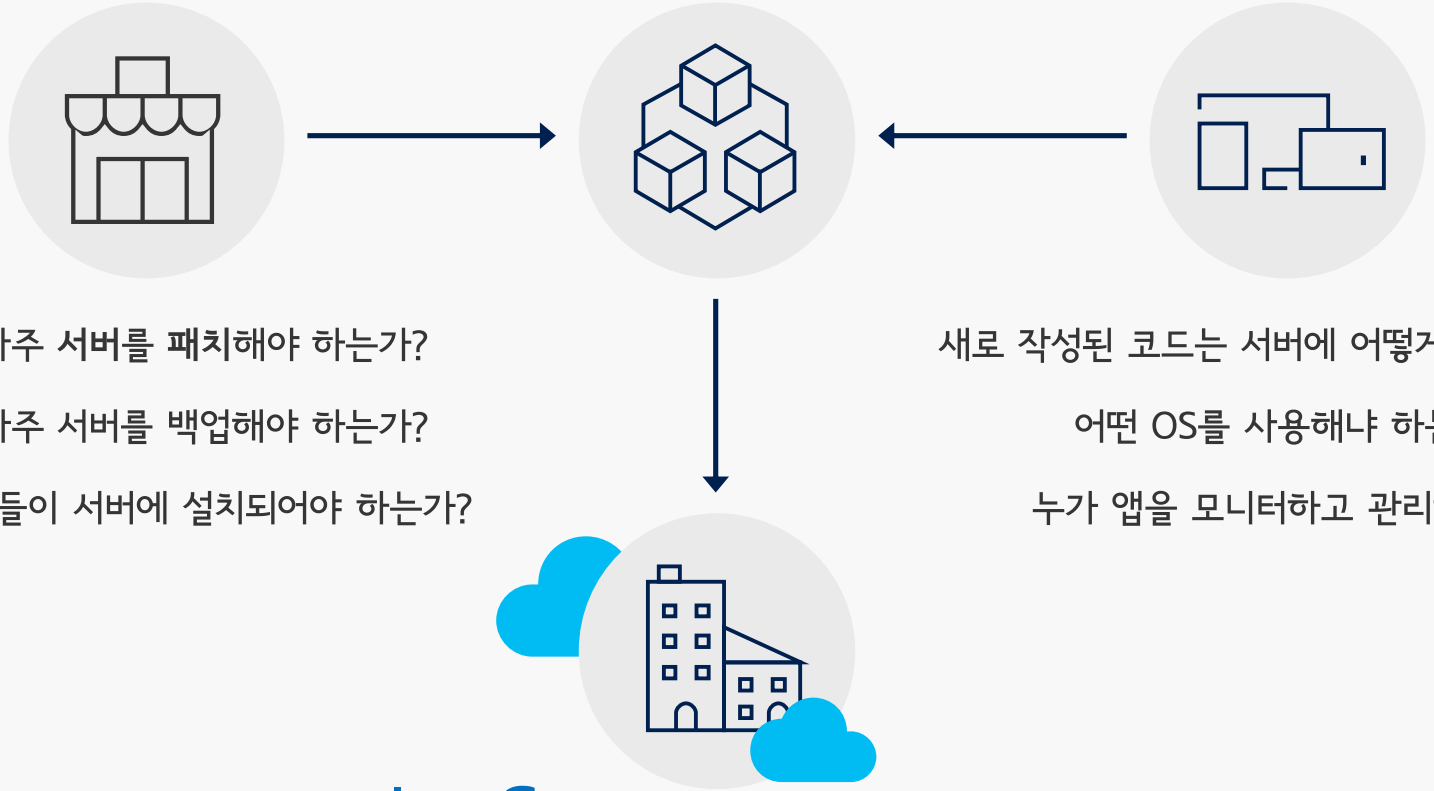
응용프로램 플랫폼의 혁명

우리 비즈니스에 **필요한 서버**의 사양은 **정확히** 어느 정도인가?

어떻게 하면 서버의 **활용도**(utilization)를 높일 수 있을까?

얼마나 **많은** 서버가 필요한가?

어떻게 우리의 앱을 **스케일**할 수 있는가?



얼마나 자주 **서버**를 **패치**해야 하는가?

얼마나 자주 서버를 백업해야 하는가?

어떤 패키지들이 서버에 설치되어야 하는가?

새로 작성된 코드는 서버에 어떻게 배포하는가?

어떤 OS를 사용해냐 하는가?

누가 앱을 모니터하고 관리하는가?

IaaS

On-Premises

응용프로램 플랫폼의 혁명

우리 비즈니스에 **필요한 서버**의 사양은 **정확히** 어느 정도인가?

어떻게 하면 서버의 **활용도**(utilization)를 높일 수 있을까?

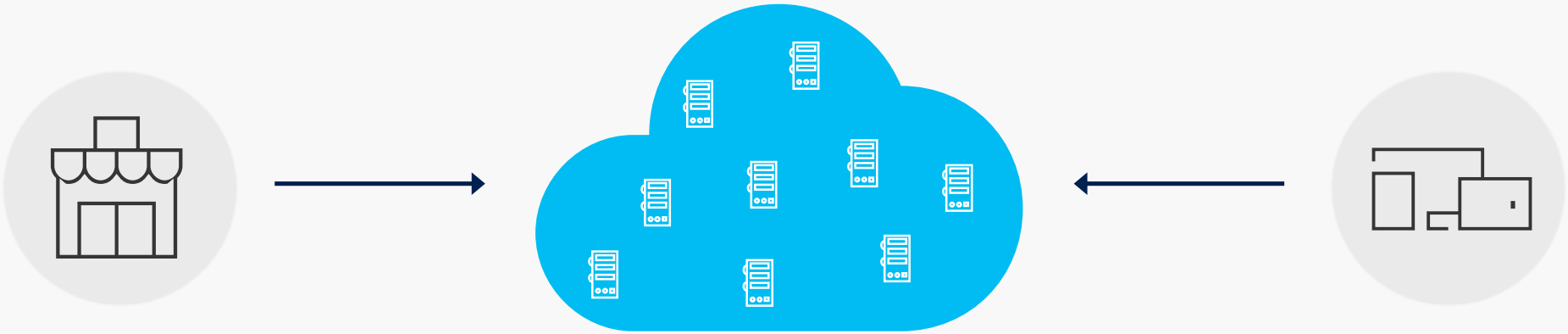
얼마나 **많은 서버**가 필요한가?

어떻게 우리의 앱을 **스케일**할 수 있는가?

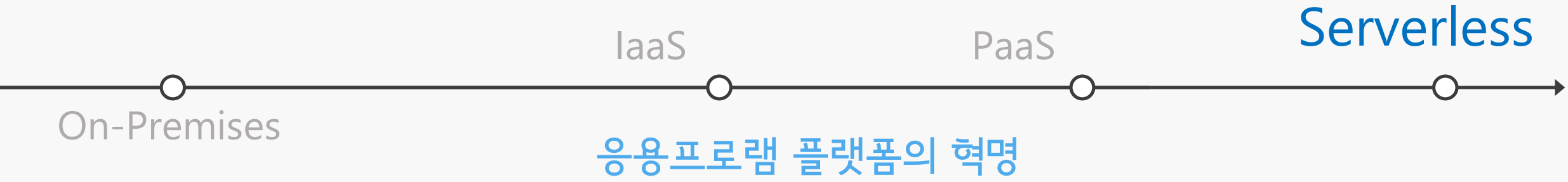
On-Premises — IaaS — PaaS

응용프로램 플랫폼의 혁명

앱을 어떻게 **설계**해야 할까?

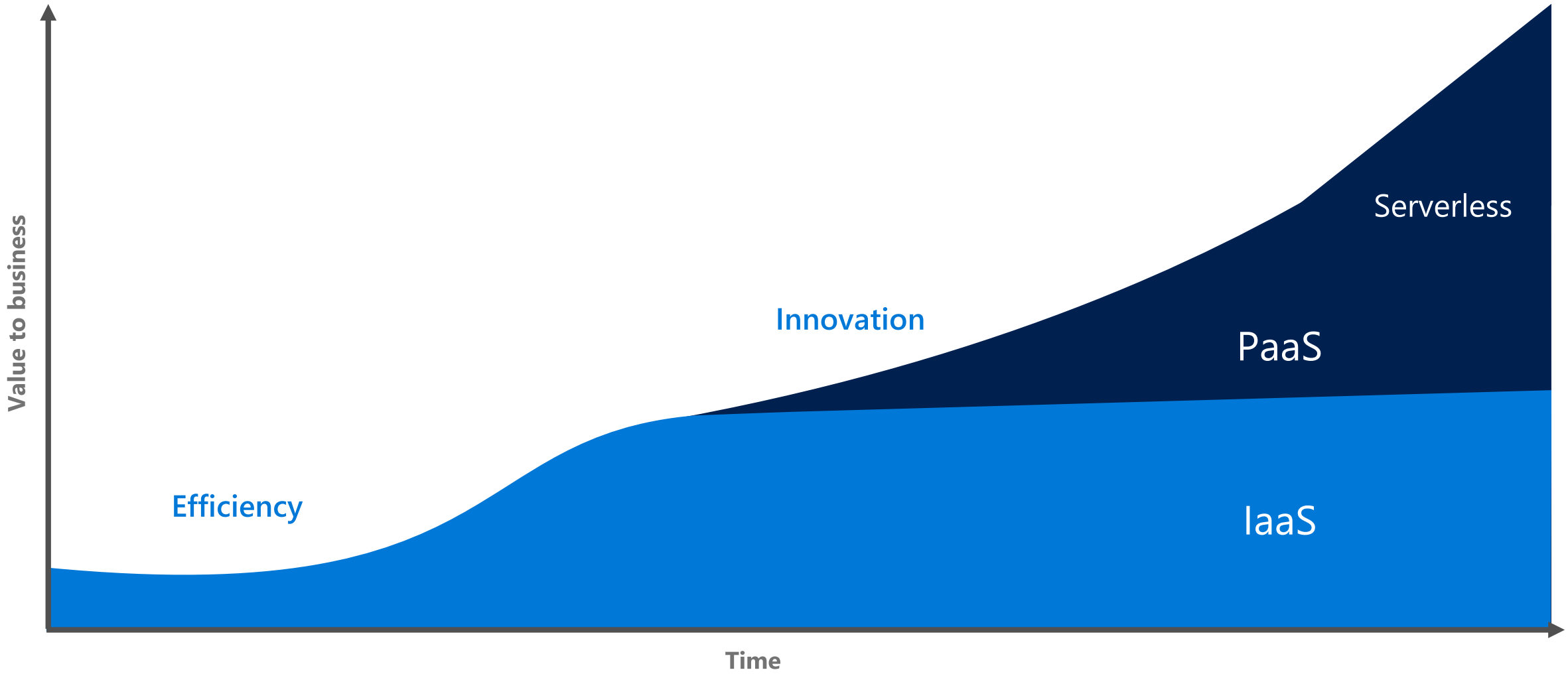


Serverless, 차세대 응용프로그램을 위한 플랫폼

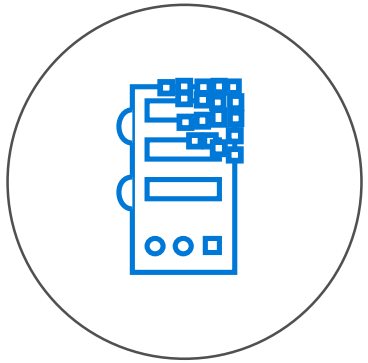


응용프로램 플랫폼의 혁명

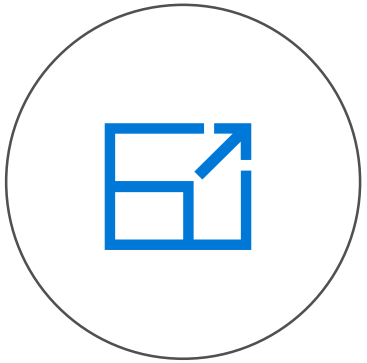
클라우드 기술의 성숙도


Value to business
Time
Efficiency
Innovation
IaaS
PaaS
Serverless

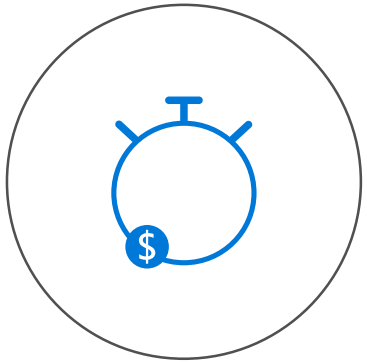
Serverless란 무엇일까?



서버의
추상화



이벤트 기반/
인스턴스 스케일

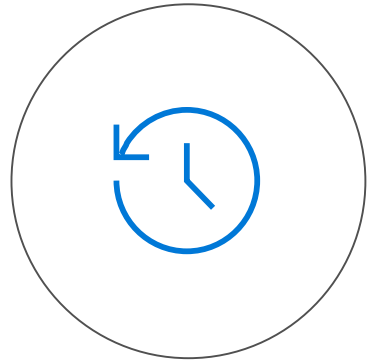


마이크로
과금

Serverless의 혜택

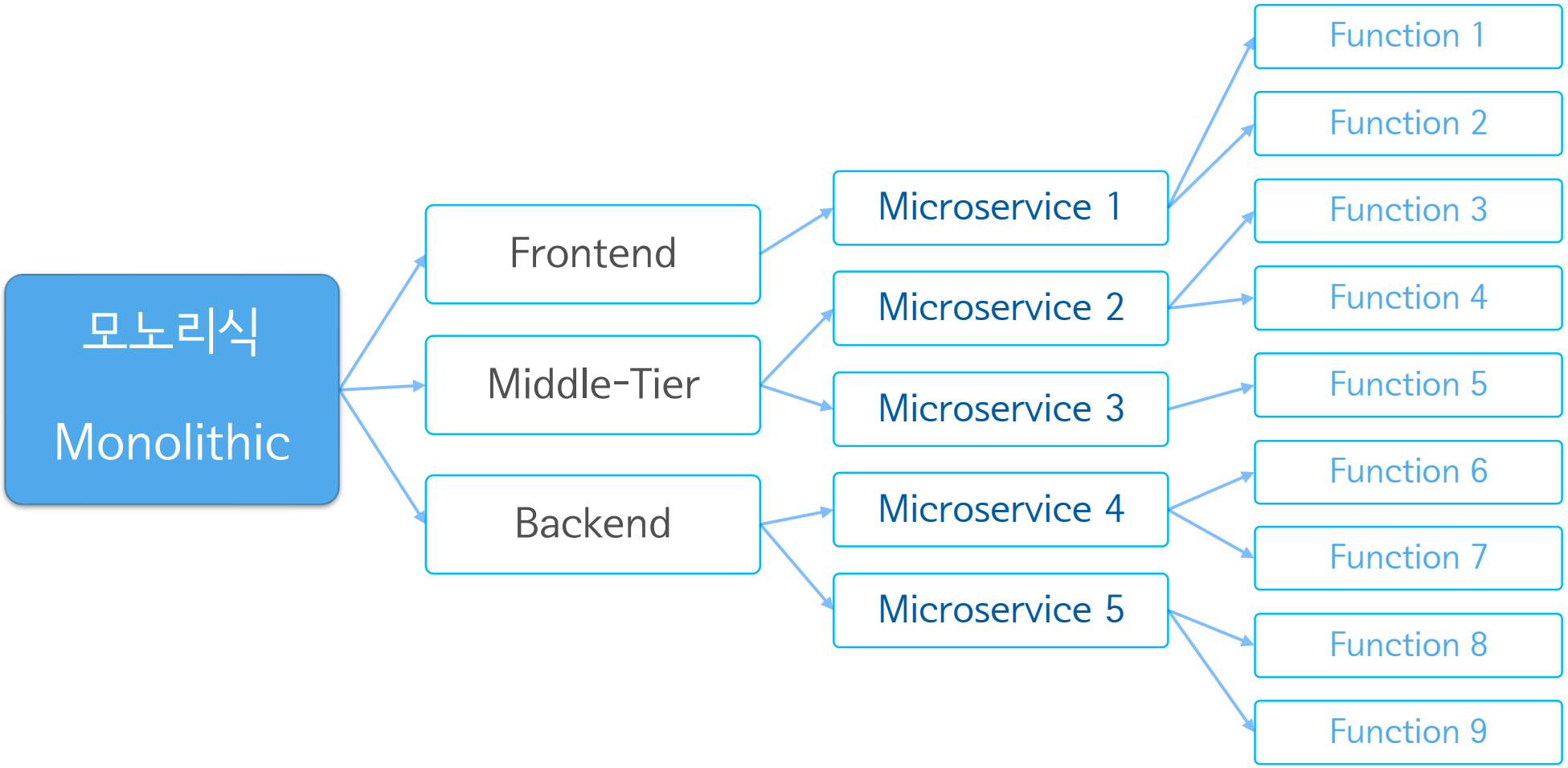
서버 관리는 무시,
앱만을 관리

빨라지는
DevOps



더 빠른
제품 출시

응용프로그램 플랫폼의 진화



서버리스(Serverless)의 의미

패턴이기 보다는 신뢰할 수 있는 시스템

서버리스 시스템에 부합하는 패턴들이 별도로 존재함

서버리스 응용프로그램

개발자가 서버를 구축하거나 관리할 필요가 없는 응용프로그램

훌륭한 PaaS 혹은 Micro-PaaS

FaaS(Function as a Service)는 서버리스의 일부

Serverless 응용프로그램 플랫폼 구성요소

개발

- IDE 지원
- 통합 DevOps
- 로컬 개발 지원
- 모니터링
- Visual 디버그 이력

플랫폼

Functions

- 개발자 생산성
- 트리거와 바인딩
- 유연한 개발 옵션들

Logic apps

- 비주얼 디자이너
- 170+ 커넥터 지원
- Functions 오케스트레이션

Event Grid

- 한 곳에서 모든 이벤트를 관리
- 거의 실시간 전달
- 광범위한 커버리지

Database

Storage

Security & Access Control

IoT

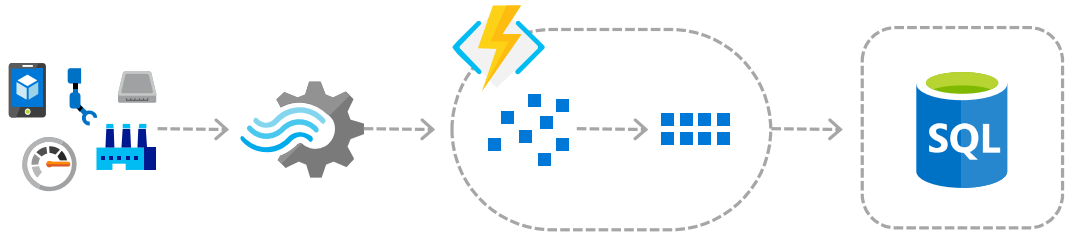
Analytics

Intelligence

서버리스 시나리오

이벤트에 응답할 필요가 있는 모든 곳에 활용할 수 있다

실시간 스트리밍 프로세싱

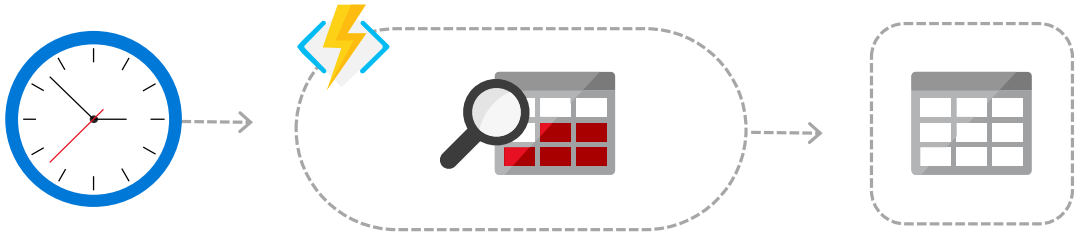


Millions of devices feed into Stream Analytics

Transform to structured data

Store data in SQL DB

시간 기반의 프로세싱



Every 15 minutes

Find and clean invalid data

Clean table

모바일 앱 백엔드

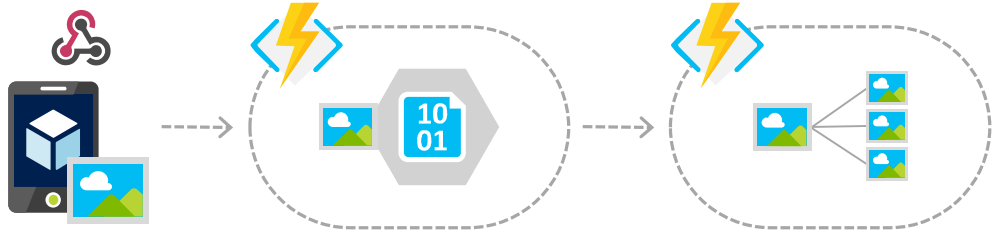
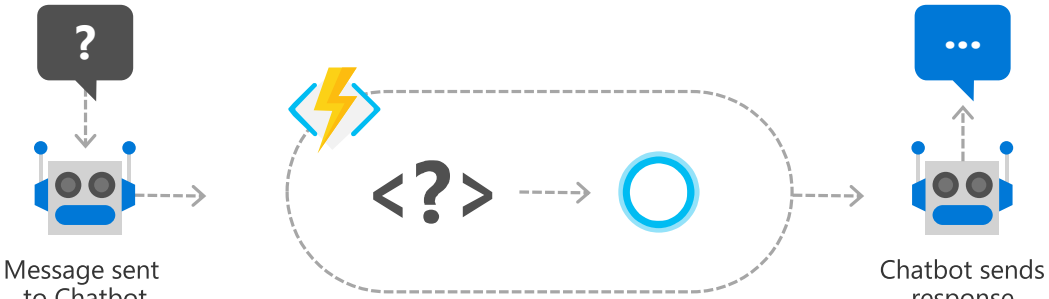


Photo taken and WebHook called

Stores in blob storage

Produces scaled images

실시간 챗봇 메시징

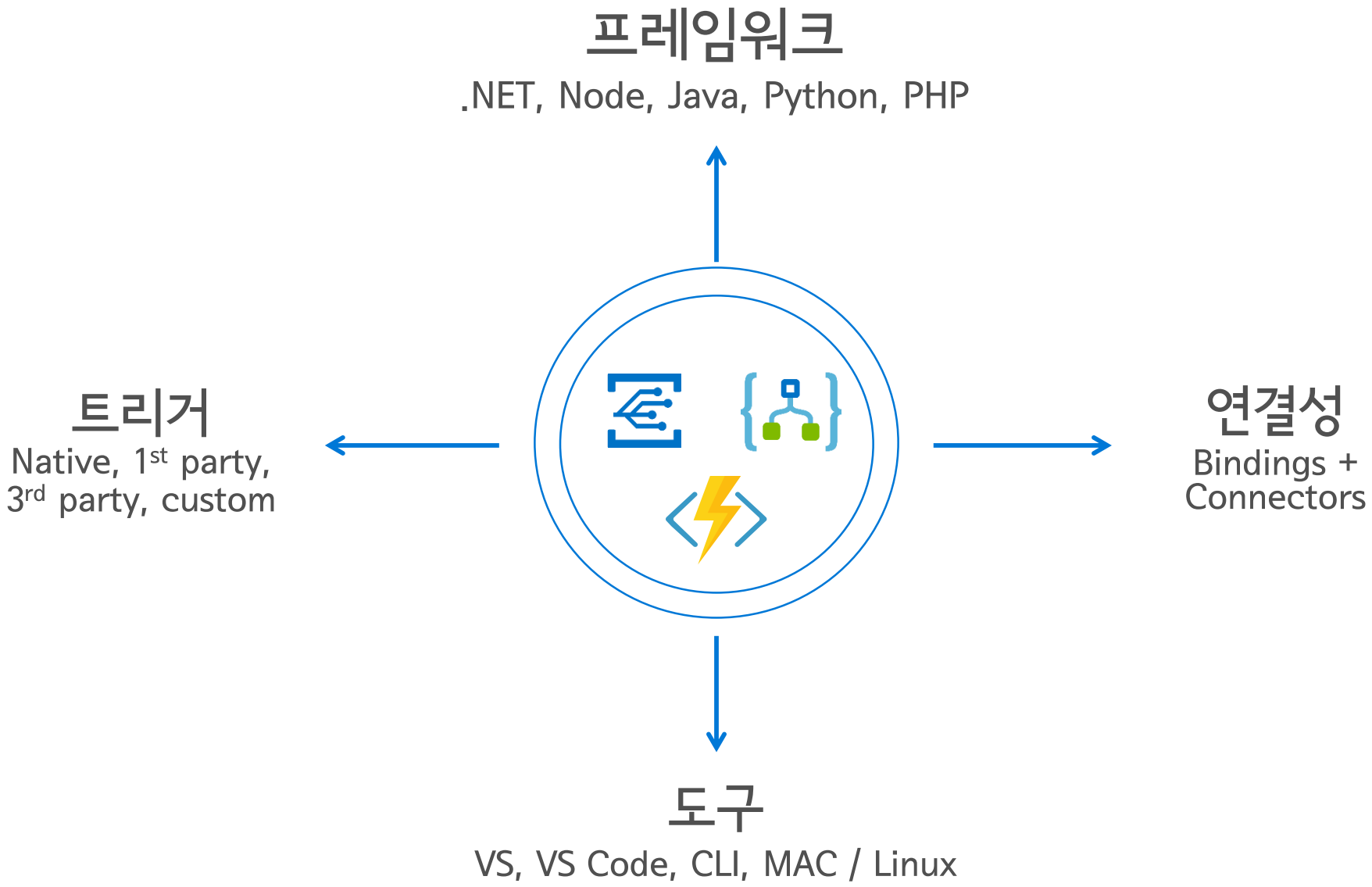


Message sent to Chatbot

Cortana Analytics answers questions

Chatbot sends response

Serverless의 기술적 모습

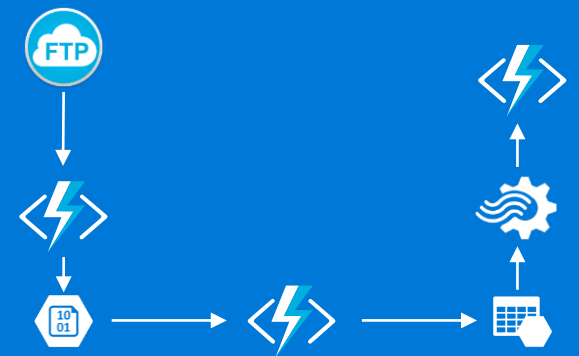


AccuWeather
SUPERIOR ACCURACY™

Real-time stream analytics

Tasks/activities

- 특정 데이터 센터에서 다른 데이터 센터로 Log들을 복제
- Log를 분석
- Data를 기반으로 액션을 수행



Before Serverless

- VM/Container/WebJobs 구성
- 빌드/패치/배포 (OS)
- 인프라(VM/Container) 모니터링
- FTP 계정 관리
- FTP 라이브러리 사용
- Azure SDK 사용

Using Serverless

- VM/Container/WebJobs 구성
- 빌드/패치/배포 (OS)
- 인프라(VM/Container) 모니터링
- 동적 설정으로 FTP 관리(자동화)
- FTP 라이브러리 사용

FUJIFILM

Tasks/activities	Before Serverless	Using Serverless
이미지 파일 관리/공유 서비스가 하루에 1테라를 처리할 수 있게 한다	온프레미스에 인프라 구성	온 프레미스에 인프라를 구성한다
높은 신뢰성과 낮은 지연시간을 제공하여 사용자 경험을 향상 시킨다	빌드/패치/배포 (OS)	빌드/패치/배포 (OS)
새로운 기능을 추가하고 더 빠르고, 더 자주 배포할 수 있다	인프라(VM/Container) 모니터링	인프라(VM/Container) 모니터링
	레거시 기술을 활용	마이크로 서비스 아키텍처를 도입하여 개발 시간은 75 % 줄였고 가동 시간은 늘었다
	서비스를 향상시키는 동안 안정성을 유지하기 어렵다	서버리스 서비스의 사용으로 지연시간이 95% 감소
	시스템 내 데이터가 증가하면 지연시간도 증가한다	


72 GB
72 GB

Azure Functions

Functions 이란

Code

Azure Functions

Events

Azure Functions

서버리스

이벤트 기반의 scale

간소화된 Dev Ops

빠른 개발

원하는 방식으로 개발

Local 개발 및 디버깅

다양한 서비스와 바인딩

Azure Service Bus

Azure Event Hub

Azure Storage

Dropbox

Sendgrid

Cosmos DB

OneDrive

Box

Twilio

Azure Function 특별한 점

다양한 개발 언어를 지원

C#, Node.js, Java, Python, F# 등

오픈 소스

https://github.com/Azure/Azure-Functions

다양한 OS를 지원

Windows, Linux, Mac OS

Container Service에서도 사용 가능

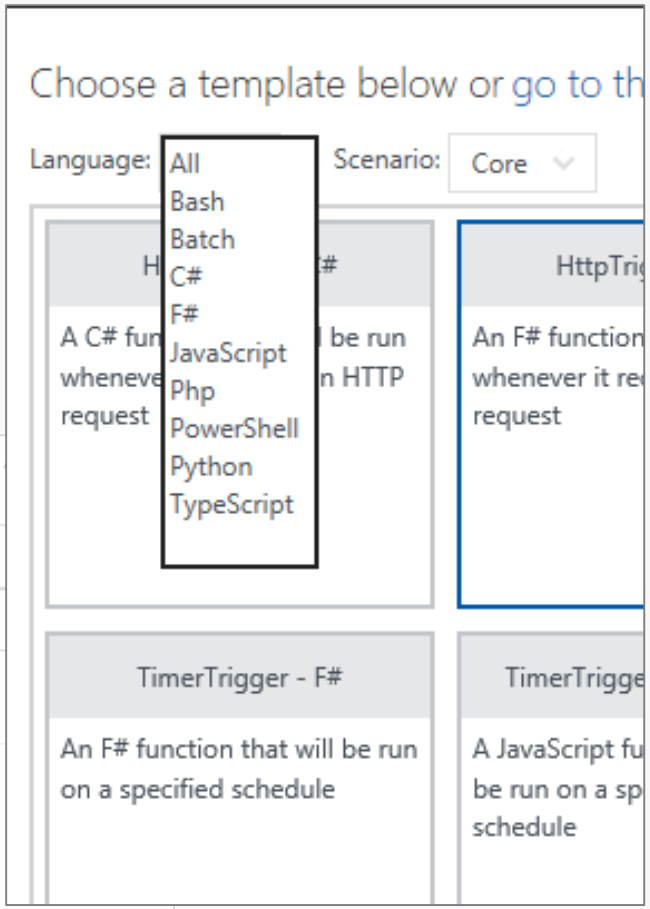
로컬 개발 및 디버깅을 지원

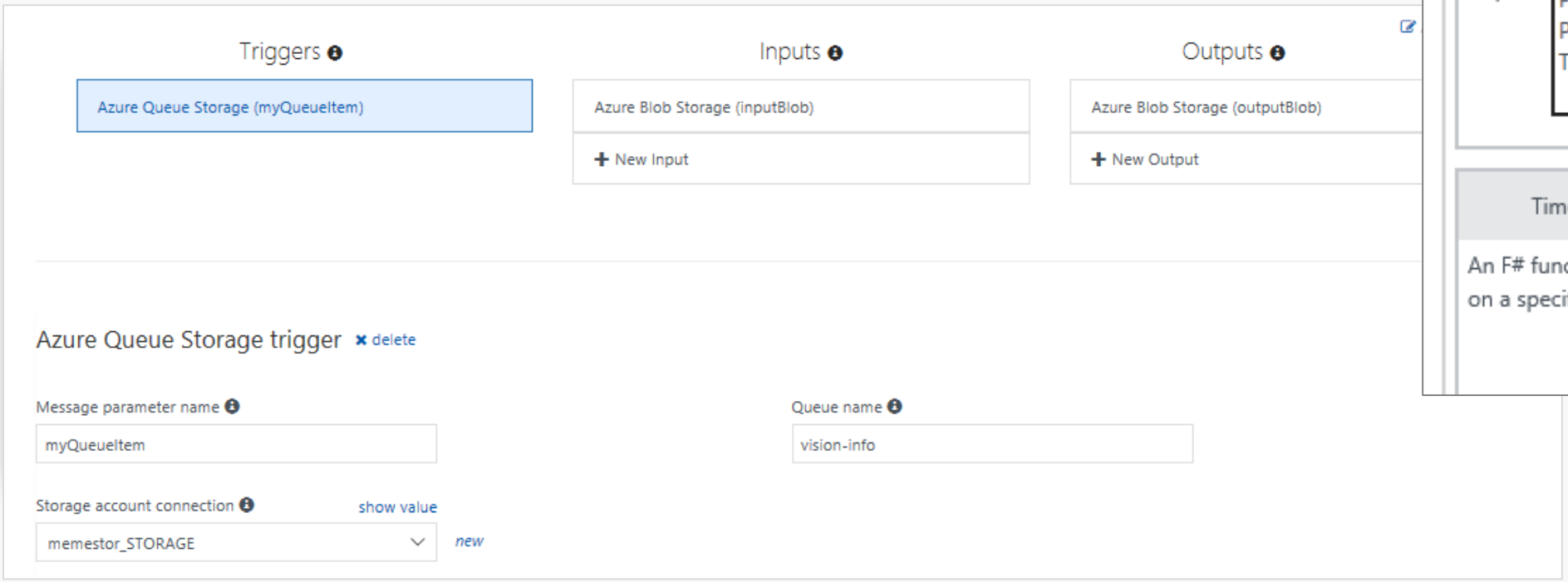
Visual Studio(Windows only)

Visual Studio Code(Linux, Mac OS, Windows)

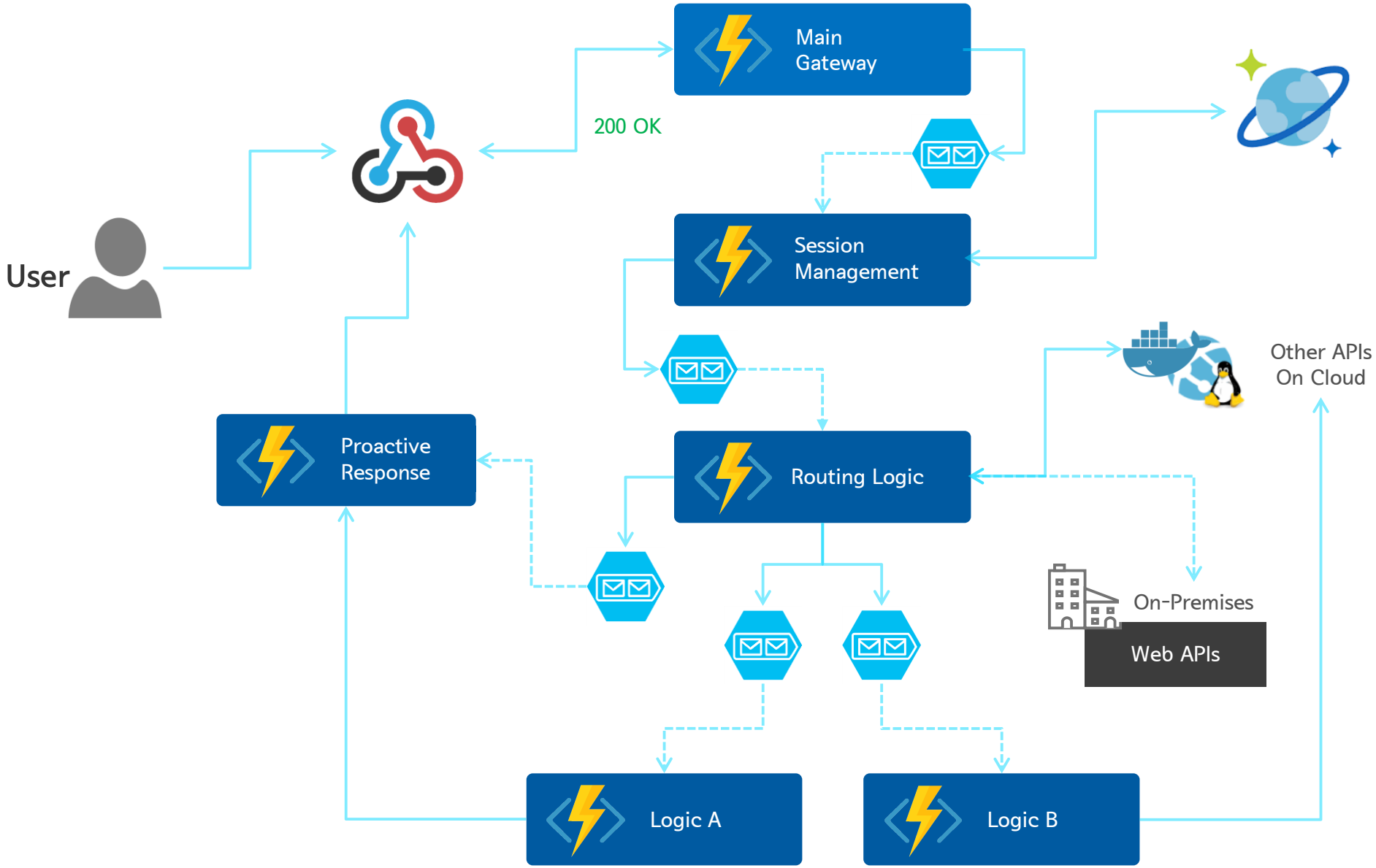
“Functions” 프로그래밍 모델

Function은 하나의 작업 단위
Function은 트리거에 의해 호출되고 실행
Function은 input, output을 갖는다



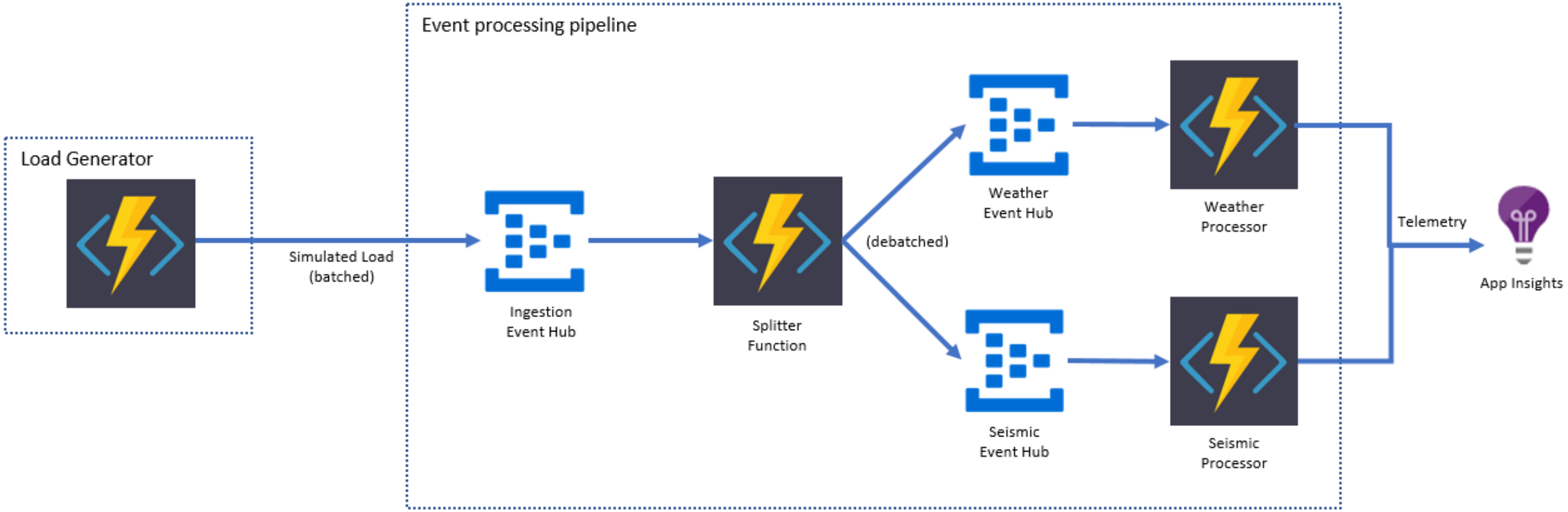


예시 : A 쇼핑몰 서버리스 예시



예시 : 초당 10 만건의 이벤트 처리 예시

Processing 100,000 Events Per Second on Azure Functions



https://blogs.msdn.microsoft.com/appserviceteam/2017/09/19/processing-100000-events-per-second-on-azure-functions/

트리거, Input, Output

(* All triggers have associated input data)+
(** The HTTP output binding requires an HTTP trigger)

Type	Service	Trigger*	Input	Output
Schedule	Azure Functions	✔		
HTTP (REST or webhook)	Azure Functions	✔		✔**
Blob Storage	Azure Storage	✔	✔	✔
Events	Azure Event Hubs	✔		✔
Queues	Azure Storage	✔		✔
Queues and topics	Azure Service Bus	✔		✔
Storage tables	Azure Storage		✔	✔
SQL tables	Azure Mobile Apps		✔	✔
NoSQL DB	Azure Cosmos DB	✔	✔	✔
Push Notifications	Azure Notification Hubs			✔
Twilio SMS Text	Twilio			✔
SendGrid email	SendGrid			✔
Excel tables	Microsoft Graph		✔	✔
OneDrive files	Microsoft Graph		✔	✔
Outlook email	Microsoft Graph			✔
Microsoft Graph events	Microsoft Graph	✔	✔	✔
Auth tokens	Microsoft Graph		✔	

코드에서 바인딩 사용하기

run.csx

```
public static void Run(byte[] image, string filename,
                       Stream outputBlob, TraceWriter log)
{
    log.Info($"Processing image: {filename}");

    var imageBuilder = ImageResizer.ImageBuilder.Current;

    imageBuilder.Build(
        image, outputBlob,
        new ResizeSettings(640, 400, FitMode.Max, null), false);
}
```

function.json

```
{
  "bindings": [
    {
      "name": "image",
      "type": "blobTrigger",
      "direction": "in",
      "path": "card-input/{filename}.jpg",
      "connection": "AzureWebJobsStorage"
    },
    {
      "type": "blob",
      "name": "outputBlob",
      "path": "card-output/{filename}.jpg",
      "connection": "AzureWebJobsStorage",
      "direction": "out"
    }
  ]
}
```

플랫폼과 Scale

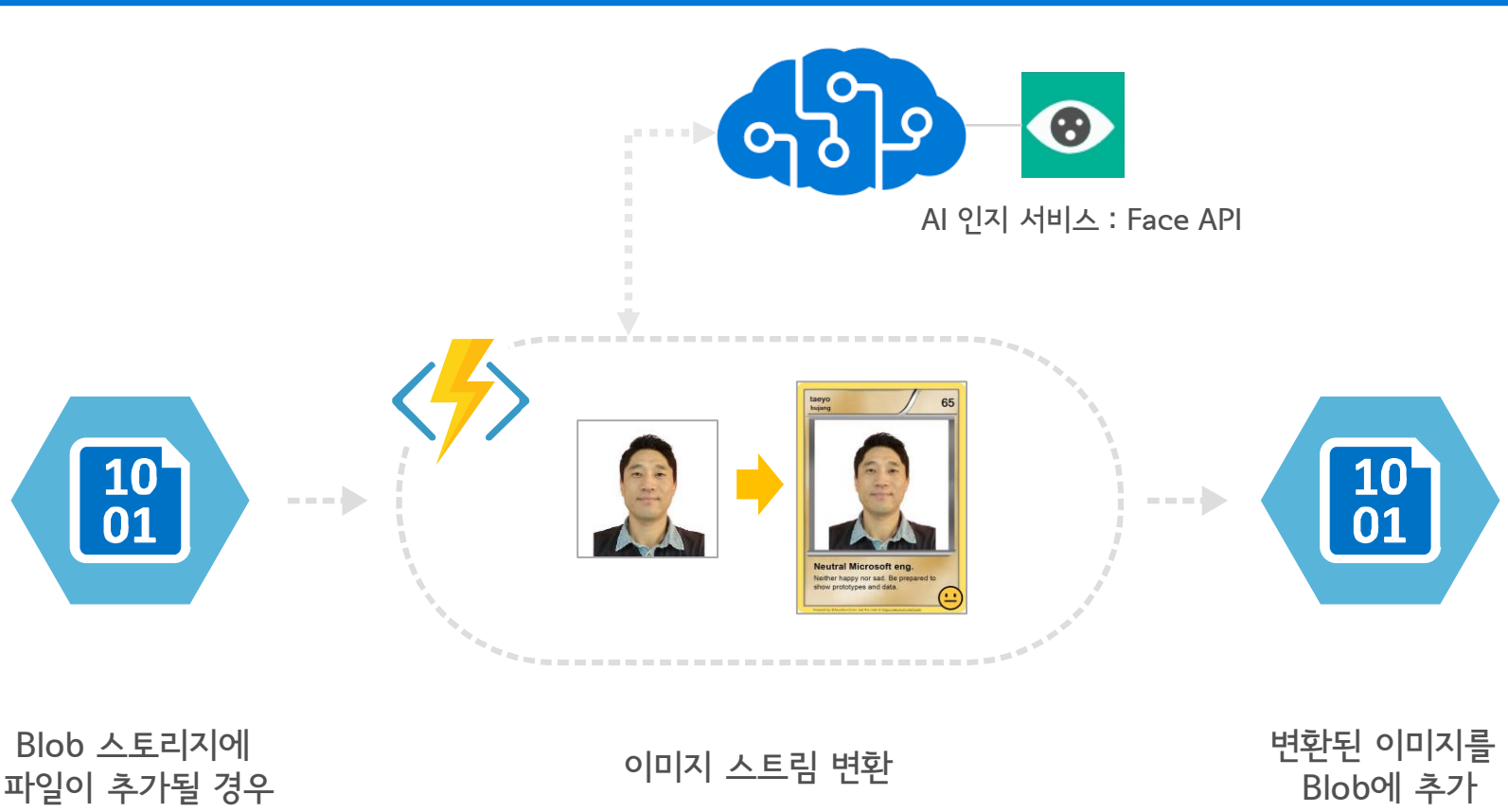
- Dedicated(전용)와 dynamic(동적) 계층 제공
- Dedicated(전용)은 기존 App Service plan을 사용
  - Basic(기본), Standard(표준), Premium(프리미엄)
  - 예약한 인스턴스(내부적으로는 VM) 숫자를 기반으로 과금
  - Scale은 운영자가 직접 관리해야 함
- Dynamic(동적)
  - 실행 수에 따라 과금
    - 실행 수 : 실행 시간 X 사용 메모리
  - Scale은 플랫폼(Azure)에서 관리함

완전한
서버리스

실습

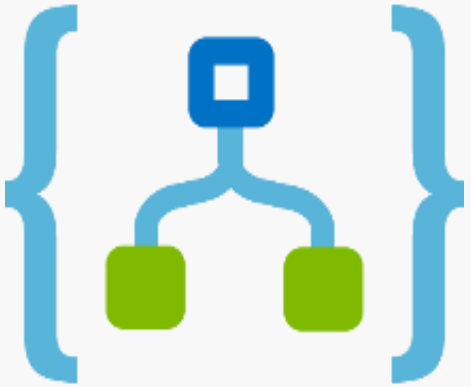
HTTP API 호출 실습 (C#, Node.js)

Blob -> Image Processing 실습 (C#)



Queue -> 메세지 처리 실습 (C#)

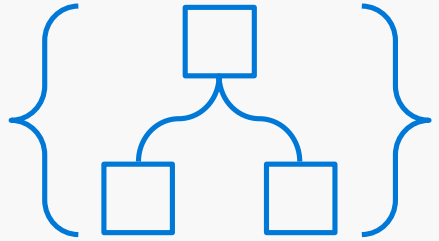
Logic App



Azure Logic Apps

Azure에서 제공되는 강력한 워크플로우 및 통합 엔진

- 비주얼 디자이너를 사용하여 트리거와 액션을 갖는 워크플로우를 빠르게 생성 및 통합할 수 있다
- 다양한 응용프로그램, 데이터, 서비스와 연결
- Azure Functions와 연결 및 오케스트레이션



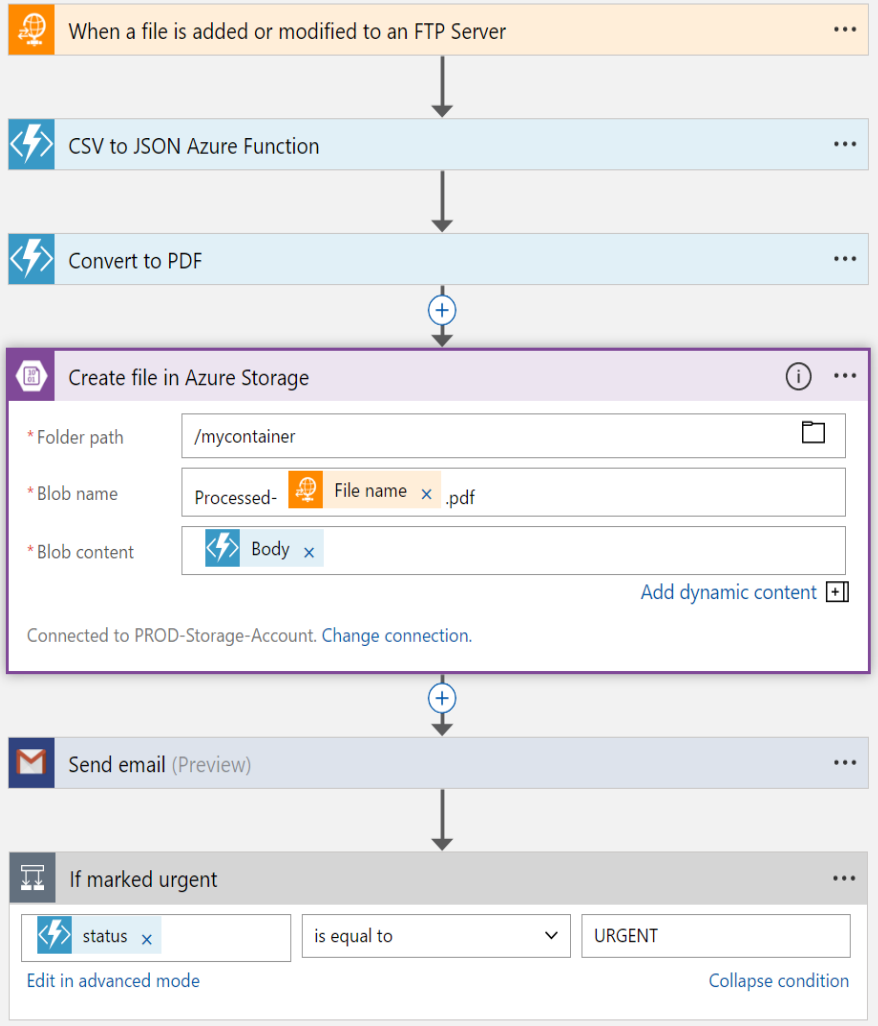
Logic Apps 워크플로우 디자이너

클라우드 기반 워크플로우

강력한 제어 흐름 제공

서로 다른 역할의 Functions과 API들을 연결

선언적으로 정의 가능. 소스 제어 체크인하거나 배포 시에 활용 가능

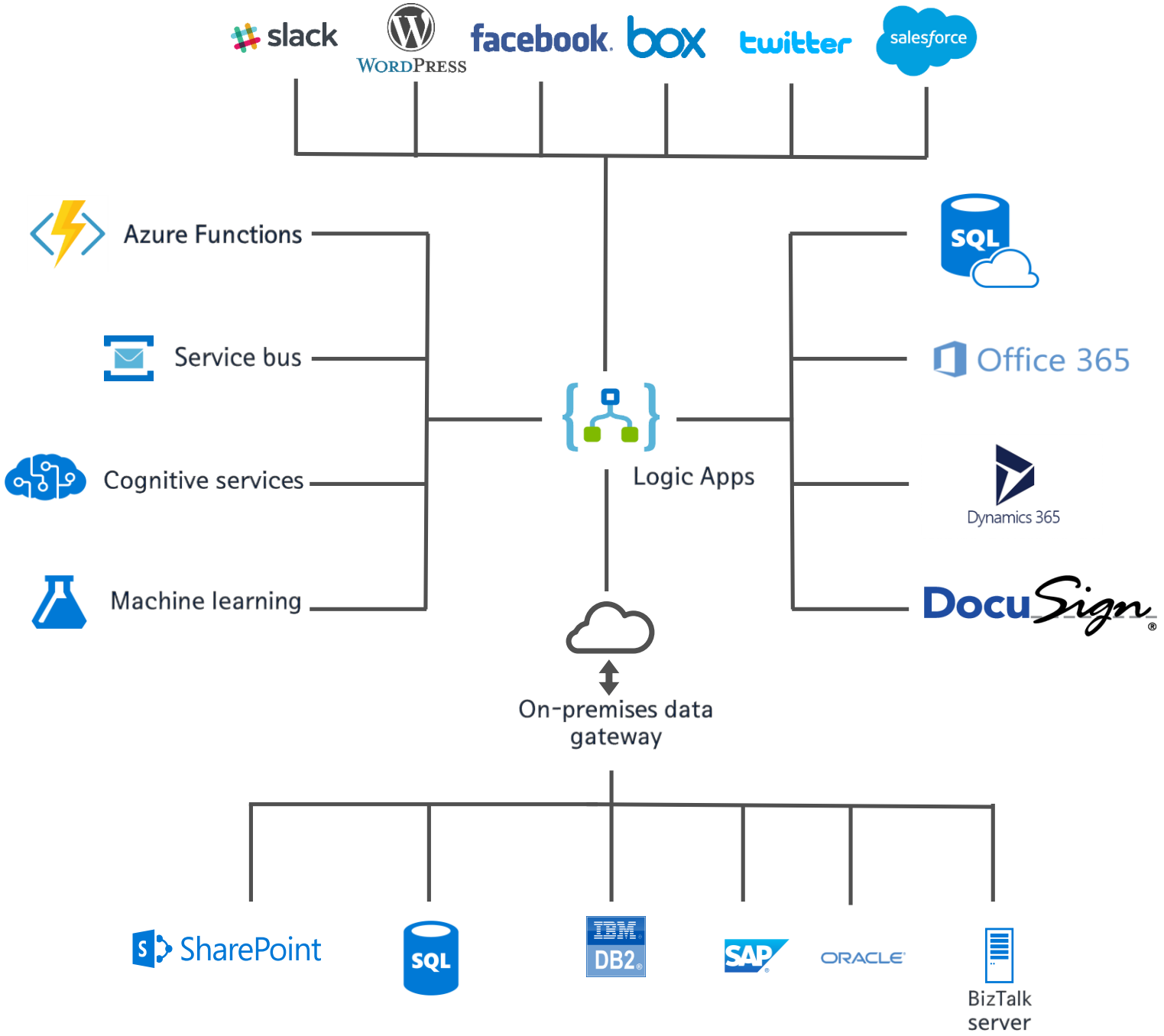


Logic Apps Connector

170개 이상 제공,
계속 늘어나는 중

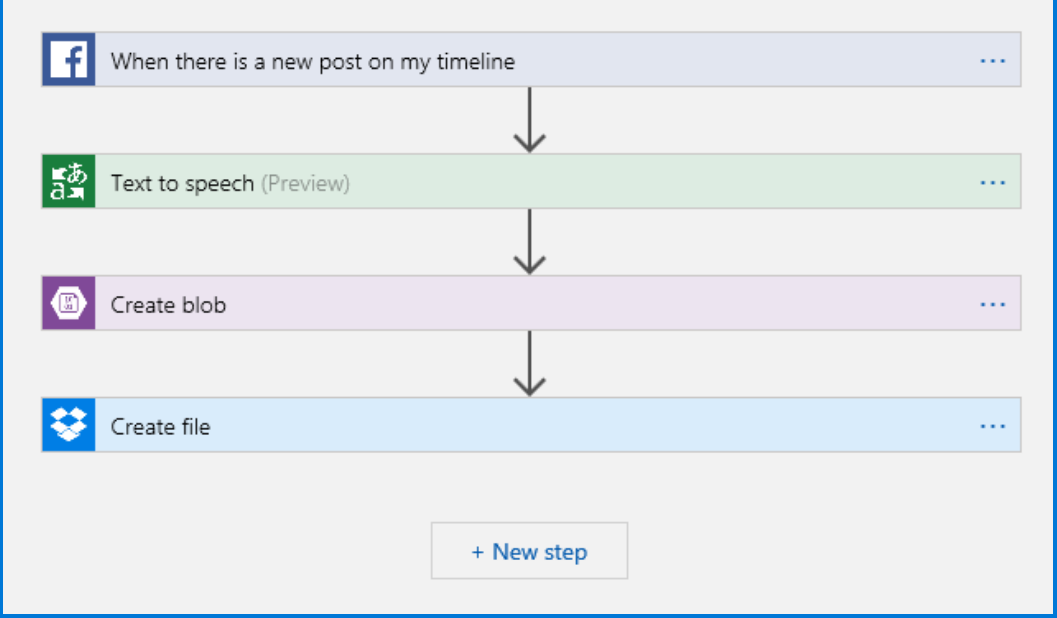
.... and more!

Logic Apps connects everything



실습

FB 메시지 >> 음성파일 변환 >> BLOB 저장 >> 메일 발송



그래서, Serverless 란

- 이벤트 기반 기술
- 인프라는 추상화
- 빠르고 규모있게 Scale
- 진짜 사용한 만큼만 과금
- 단, 아키텍처에 대해서는 고민을 좀 해봐야 한다
- Azure는 훌륭한 Serverless 포트폴리오를 제공한다

Azure 서버리스를 시작하려면

Try Functions - https://functions.azure.com/try
Try App Service - https://tryappservice.azure.com