Welcome

Azure Development

Nagaraju B
Sr. Program Manager
Development Head - .NET

Microsoft Certified Trainer
Microsoft Certified Solution Architect
http://nbende.wordpress.com

Twitter
@nbende

FaceBook
nbende

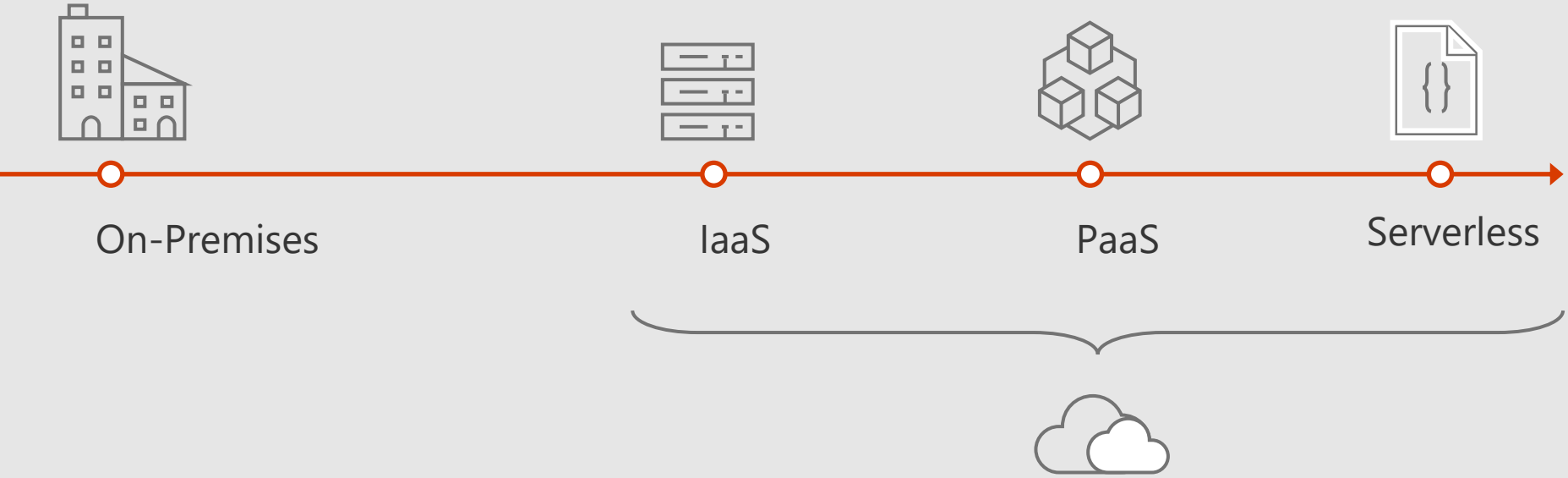
Agenda

Serverless Computing
Azure Functions
Logic Apps
Other

Serverless Computing

Serverless computing promises agility & power in building next generation of solutions

The evolution of application platforms



Data center

Virtual machines

IaaS

PaaS

Serverless

More host flexibility
Less deployment flexibility
Capital expense

Less host flexibility
More deployment flexibility
Operational expense

Before cloud

What is the right **size** of **servers** for my business needs?

What happens in case of **server hardware** failure?

How do I **deploy** new **code** to my **server**?

What media should I use to keep backup?

Which packages should be on my **server**?

How can I increase **server** utilization?

How can I **scale** my app?

What size of **servers** should I **buy**?

Who **monitors** my **App**?

How often should I backup my **server**?

Who **monitors** my **Servers**?

What storage I need to use?

Do I need secondary network connection?

How many **servers** do I need?

Who has **physical** access to my **servers**?

Which OS should I use?

Do I need a UPS?

What happens if the power goes out?

Are my **server** in a secure location?

It takes how long to **provision** a new **server**?

How often should I **patch** my **servers**?

How can I dynamically configure my app?

Then came IaaS—table stakes for digital business

What is the right **size** of **servers** for my business needs?

How can I increase **server** utilization?

How many **servers** do I need?

How can I **scale** my app?

How often should I **patch** my **servers**?

How often should I backup my **server**?

Which packages should be on my **server**?

How do I **deploy** new **code** to my **server**?

Which OS should I use?

Who **monitors** my App?

Then PaaS, critical for digital transformation

What is the right **size** of **"servers"** for my business needs?

How can I increase **"server"** utilization?

How many **"servers"** do I need?

How can I **scale** my app?

What is Serverless?

Abstraction
of servers

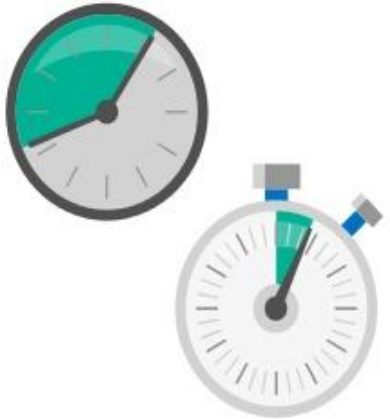
Event-driven/
instant scale

Sub-second
billing

Benefits of Serverless?

Reduced DevOps

Focus on Business Logic



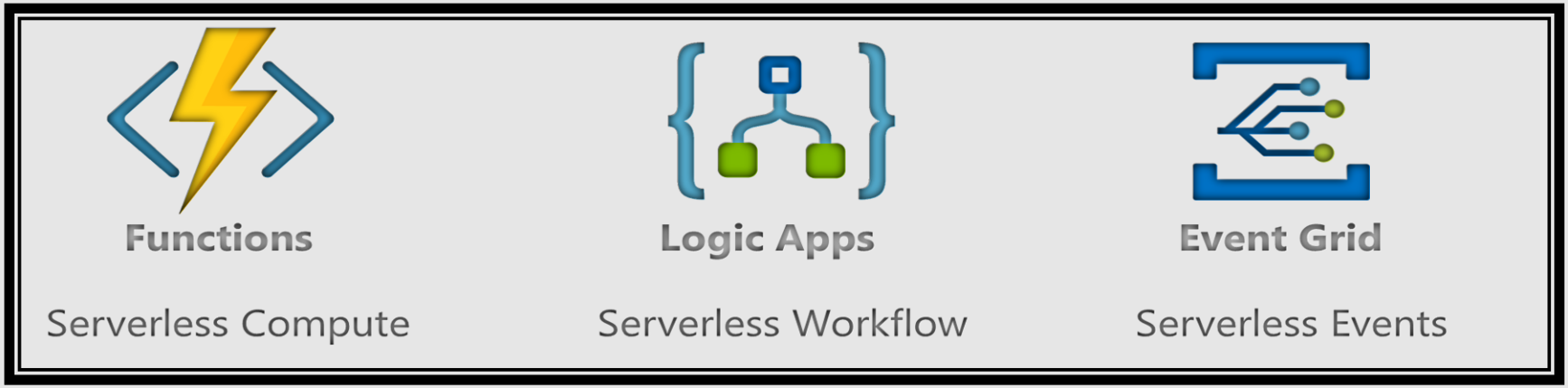
Reduced Time To Market

- Serverless is:
  - Referred to as Function-as-a-Service (FaaS) instead of Platform-as-a-Service (PaaS)
  - The cloud provider manages starting and stopping instances
  - The instances run only to serve a request and then they immediately shut down
  - Requests are typically billed abstractly by measuring consumption and usage time
- Benefits:
  - Theoretically infinite scale driven by events
  - Applications only contain business logic, not hosting logic
  - Faster time-to-market
  - Abstraction of servers
  - Sub-second billing

Azure offerings for serverless computing

- Azure Functions
- Azure Logic Apps

combination of these services is the best way to rapidly build a scalable, full-featured integration solution



Introducing Azure Functions

Azure Functions

Code

+

Events + data

Function Apps

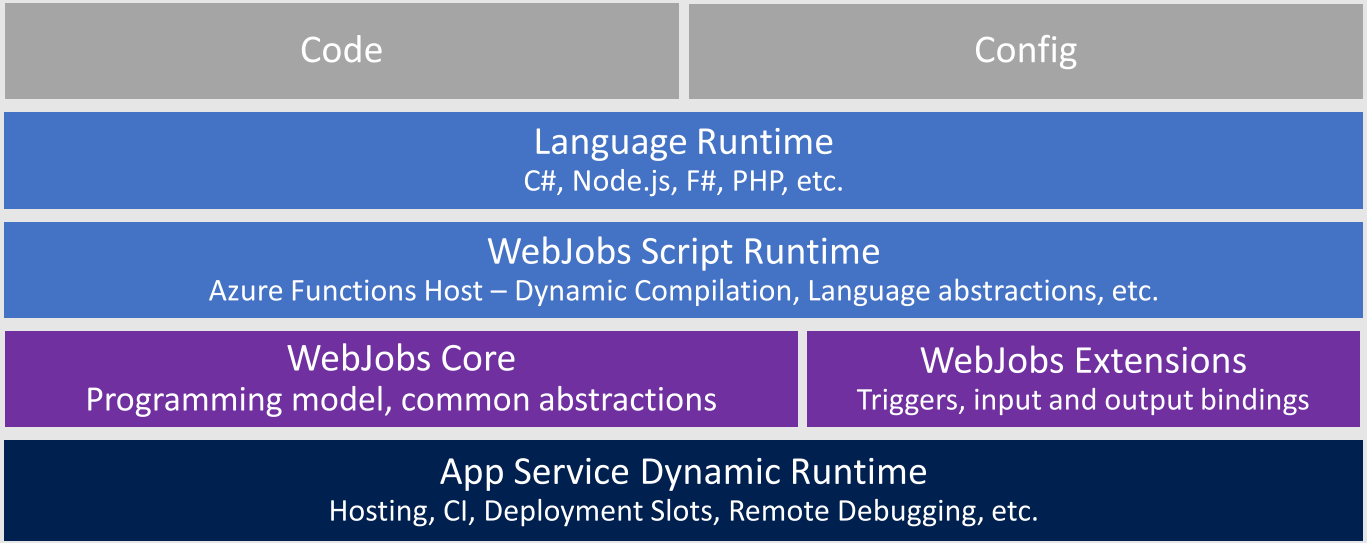
- Reduces friction to get code running in the cloud
- Allows development in both application stacks (.NET, Node, Python, etc.) and scripting platforms (PowerShell, Bash, etc.)
- Exposed as HTTP endpoints
- Integrates with Logic Apps
- Scale based on event-driven demand:
  - React to changes in a database
  - Create time-based events
  - Trigger based on an HTTP Request
  - Trigger based on Event Grid and an Azure event

Function Apps

- Methods of Execution:
  - Triggers
  - Webhooks
  - Timer
- Language of Choice:
  - C#, F#, Node.js, Python, PHP, batch, bash, Java, or any executable
- Flexible Pricing:
  - Dynamic (pay-per-use)
  - App Service Plan
- Integrations:
  - DocumentDB, Event Hubs, Mobile Apps (tables), Notification Hubs, Service Bus, Storage
  - GitHub (webhooks), On-premises (using Service Bus)

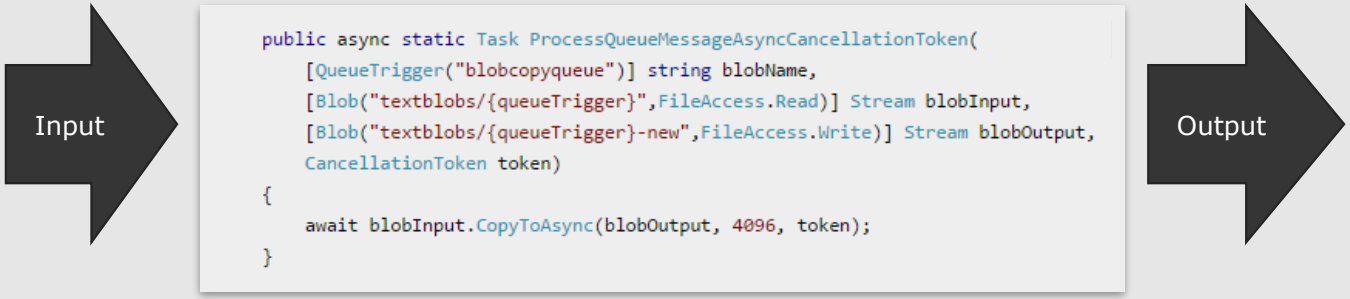
Architecture

Built on top of App Service and WebJobs SDK

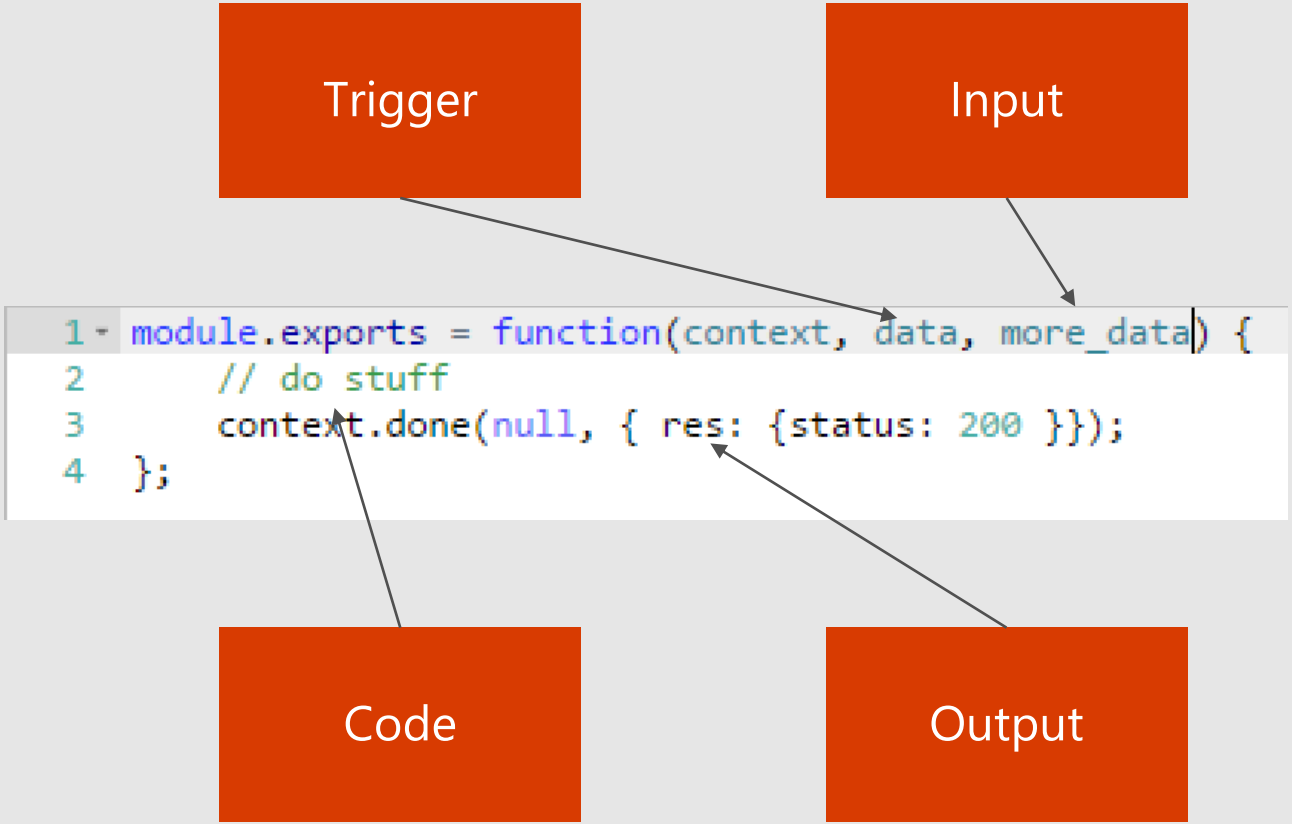


Integrations

- Function as the unit of work
- Functions are executed; they start and finish
- Functions have inputs and outputs

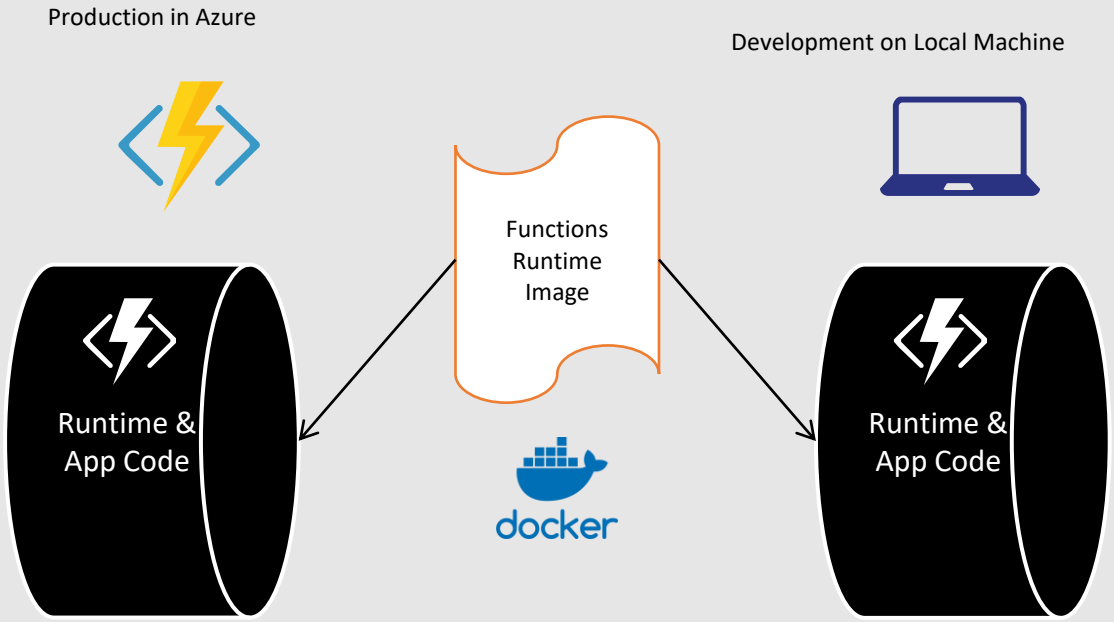


Programming Concepts



Function Apps on Linux

Docker Hub: microsoft/azure-functions-runtime



Serverless Computing (FaaS)

- Function as a Service (Azure Functions)
- Platform to develop, run, and manage application
- Without the complexity of building and maintaining the infrastructure
- Azure Functions can also be a delivery mechanism for microservices

Similar Technologies

- AWS Lambda
- Google Cloud Functions
- Open Whisk (IBM)

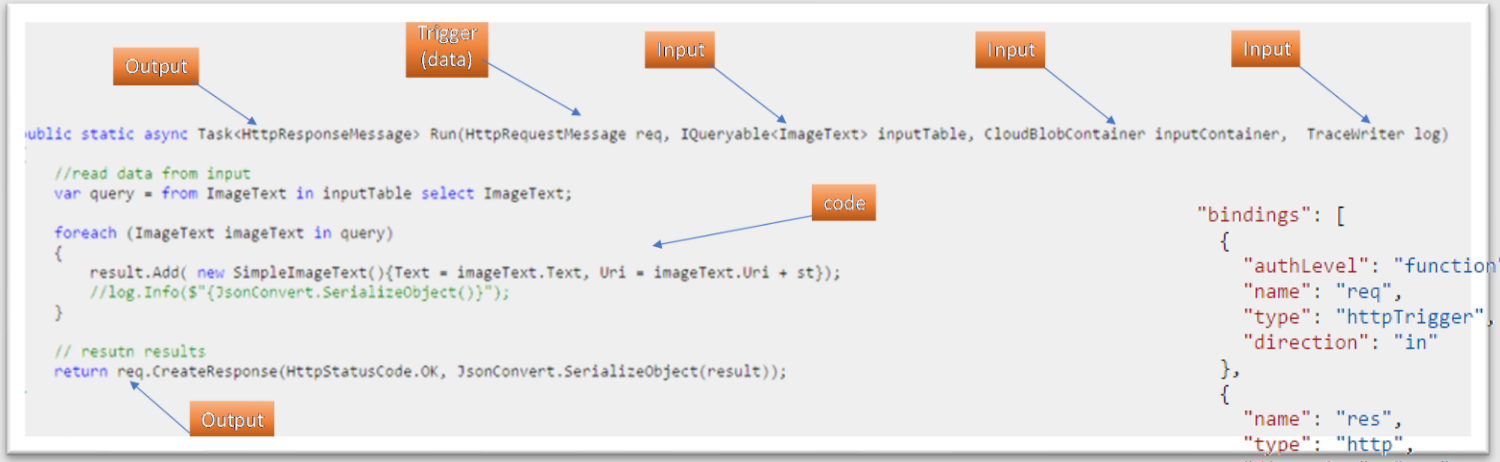
Triggers and Bindings

Type	Service	Trigger	Input	Output
Schedule	Azure Functions	✓		
HTTP	Azure Functions	✓		✓
Blob Storage	Azure Storage	✓	✓	✓
Events	Azure Event Hubs	✓		✓
Queues	Azure Storage	✓		✓
Queues and topics	Azure Service Bus	✓		✓
Storage tables	Azure Storage		✓	✓
No-SQL DB	Azure CosmosDB		✓	✓
Push notifications	Azure Notification Hubs			✓
Twilio SMS Text	Twilio			✓
SendGrid email	SendGrid			✓

Automatically referenced packages

- These assemblies are automatically referenced:
  - `mscorlib`
  - `System`
  - `System.Core`
  - `System.Xml`
  - `System.Net.Http`
  - `Microsoft.Azure.WebJobs`
  - `Microsoft.Azure.WebJobs.Host`
  - `Microsoft.Azure.WebJobs.Extensions`
  - `System.Web.Http`
  - `System.Net.Http.Formatting`

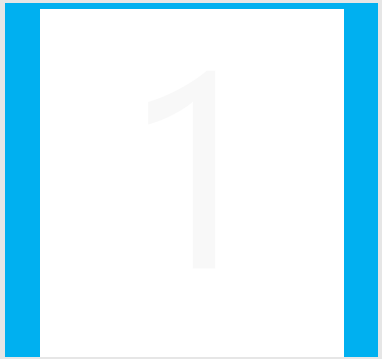
Functions Programming Model

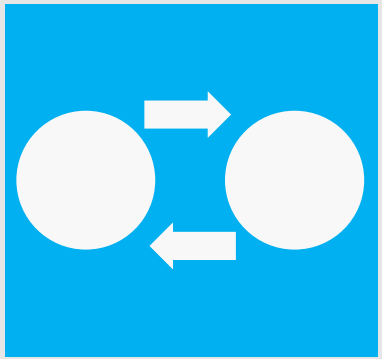


- Function as a single unit of work
- Functions are executed per trigger
- Functions have inputs and outputs

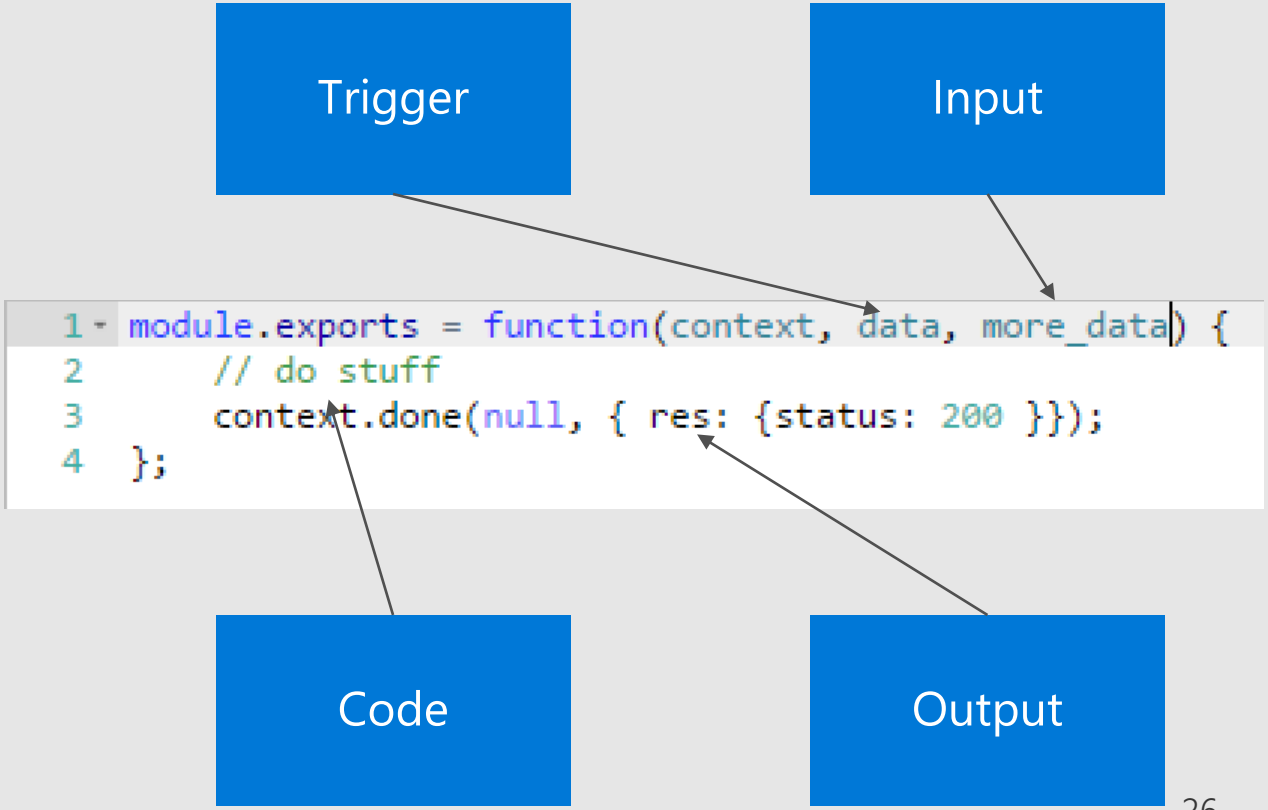
Best practices for the “Functions” model

- Functions *should* “do one thing”
- Functions *should* be idempotent
- Functions *should* finish as quickly as possible

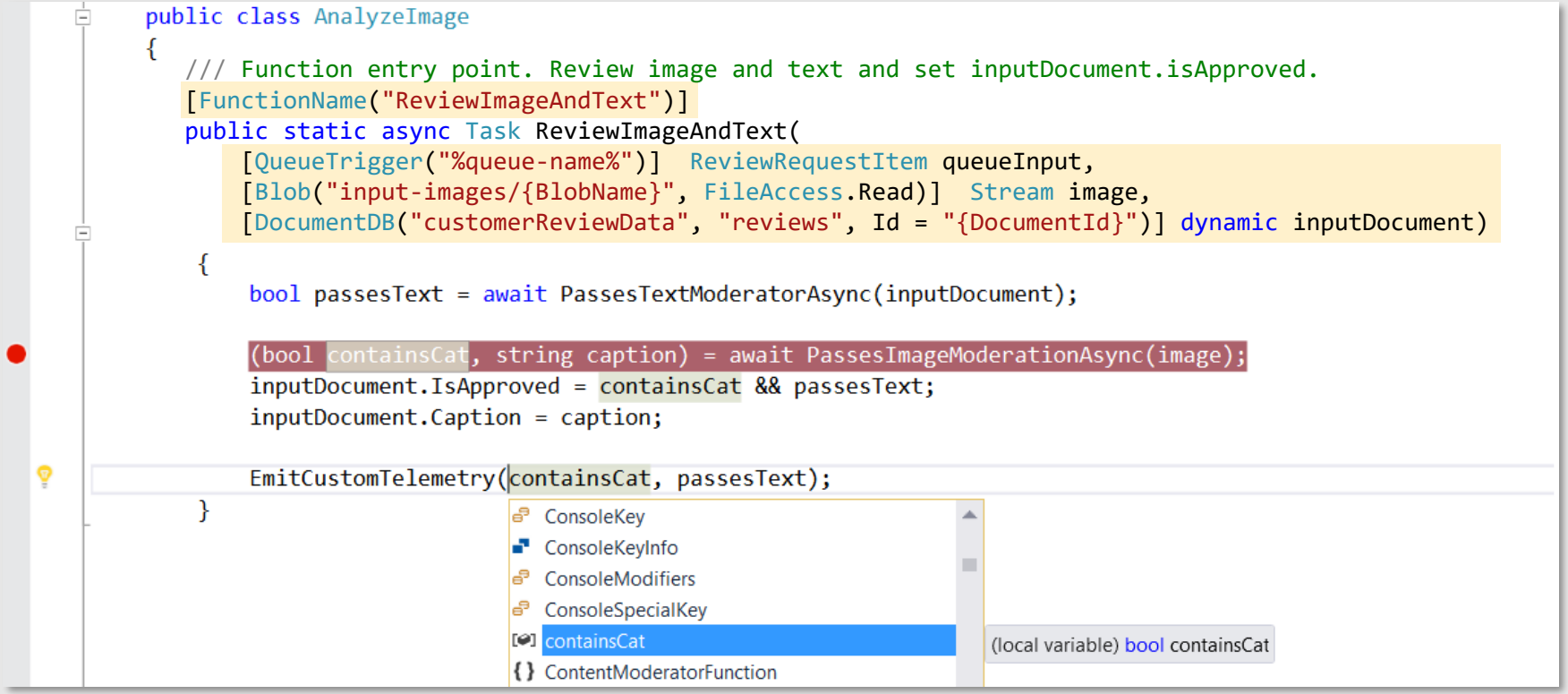




Functions programming concepts



Visual Studio Tooling



VSTS

- Build and deploy with VSTS

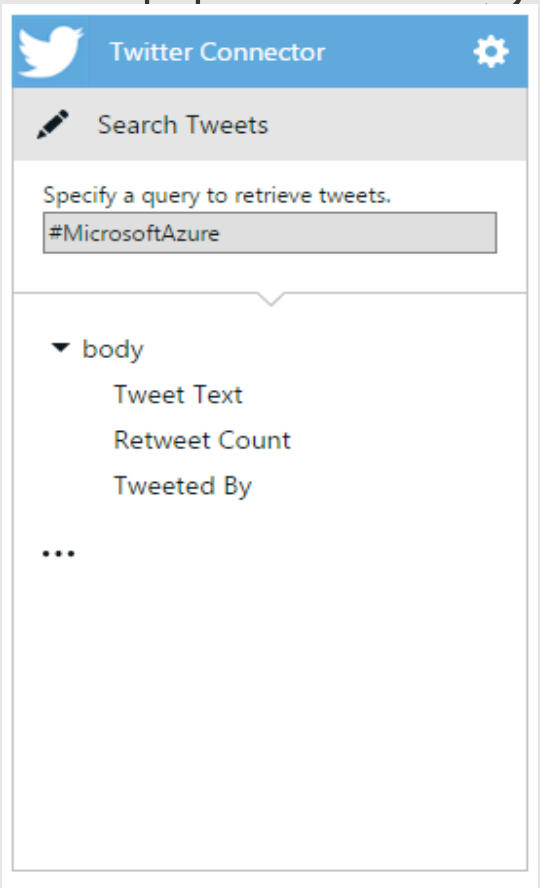
Demos

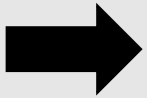
Resources

- Microsoft Docs – https://docs.microsoft.com
- Azure Functions – https://functions.azure.com
- GitHub Repo – https://aka.ms/func-github
- Twitter - @AzureFunctions

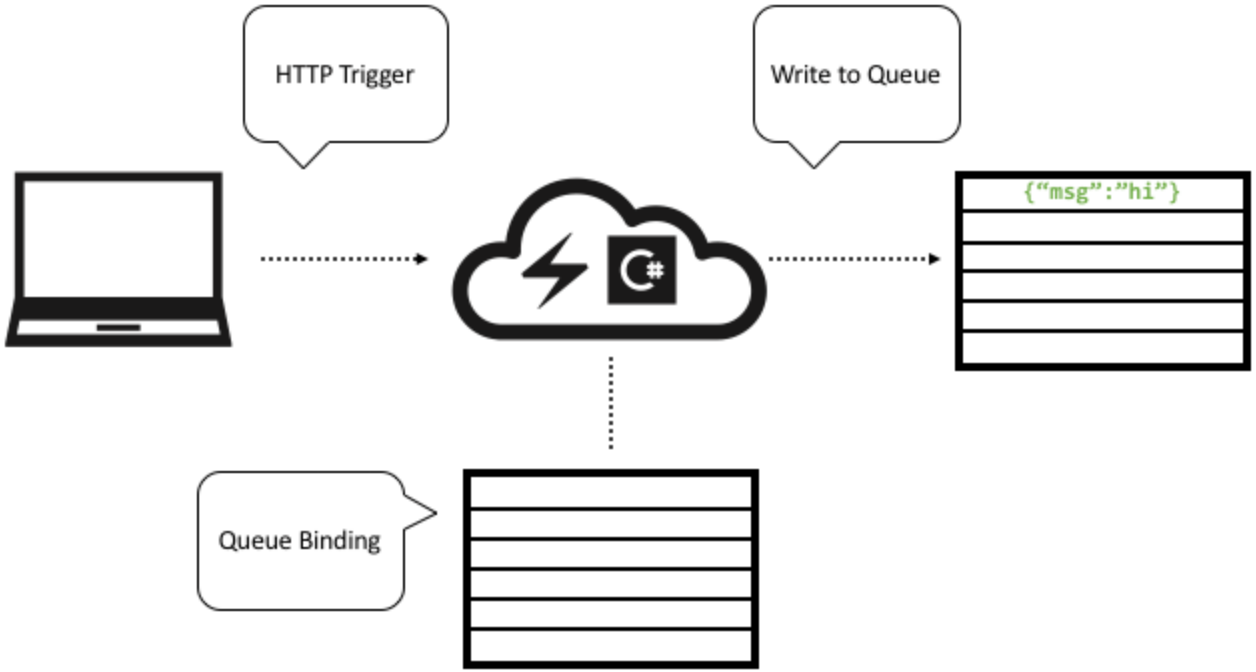
Q&A

Function Apps in Logic Apps





Custom Function App


HTTP Trigger
Write to Queue
{“msg”:”hi”}
Queue Binding

Programming Model Differences

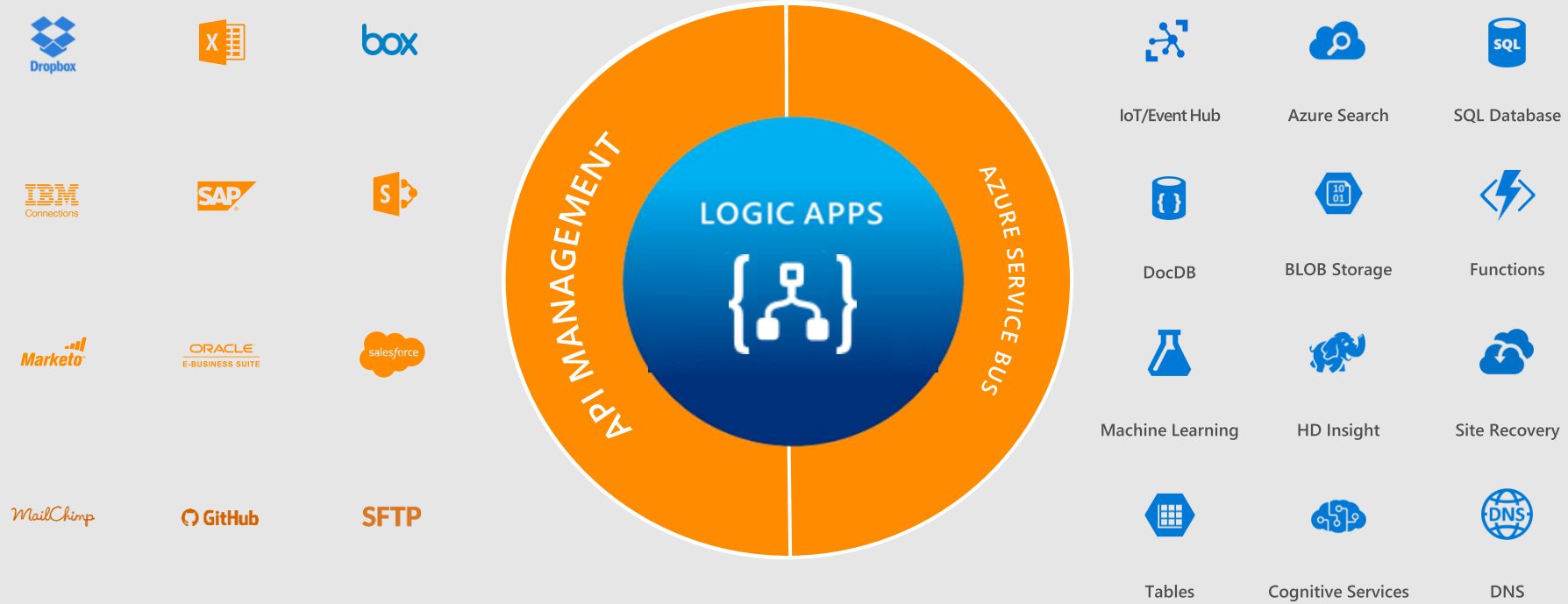
WebJobs SDK

- ✓ C# only
- ✓ Attributes for configuring bindings
- ✓ Traditional .NET developer experience (Visual Studio, NuGet, MSBuild)
- ✓ Many functions per class
- ✓ Can access and manipulate many core SDK features
- ✓ Can’t listen for HTTP requests*

Azure Functions

- ✓ C# & Node.js + more
- ✓ Config files for configuring bindings
- ✓ More diverse development experience (Web portal, VSCode, dynamically builds itself)
- ✓ Limited access to manipulate core SDK features, but (C# only) still some access
- ✓ Supports HTTP!

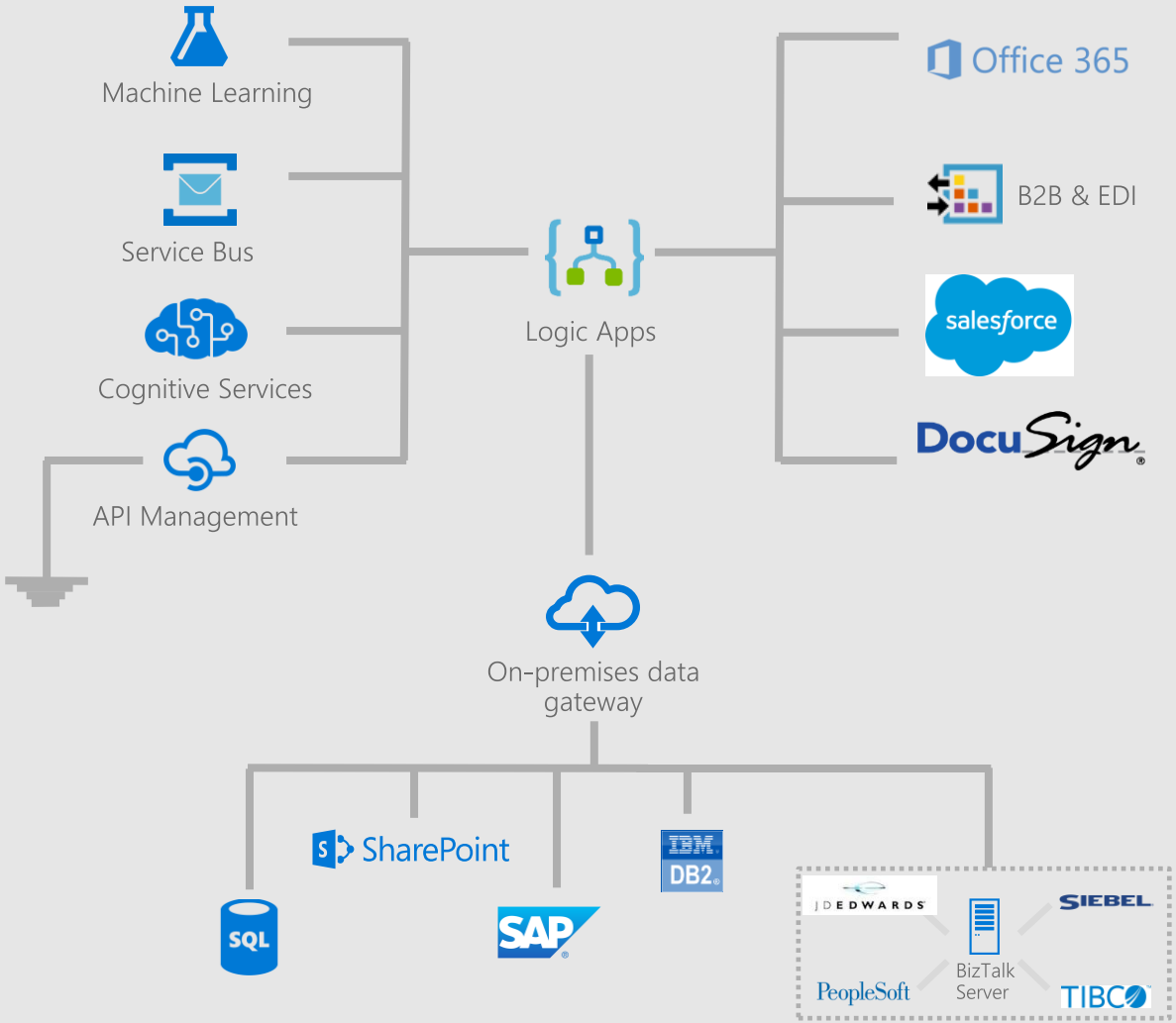
Microsoft's Cloud Integration Platform



Hybrid

Secure Gateway

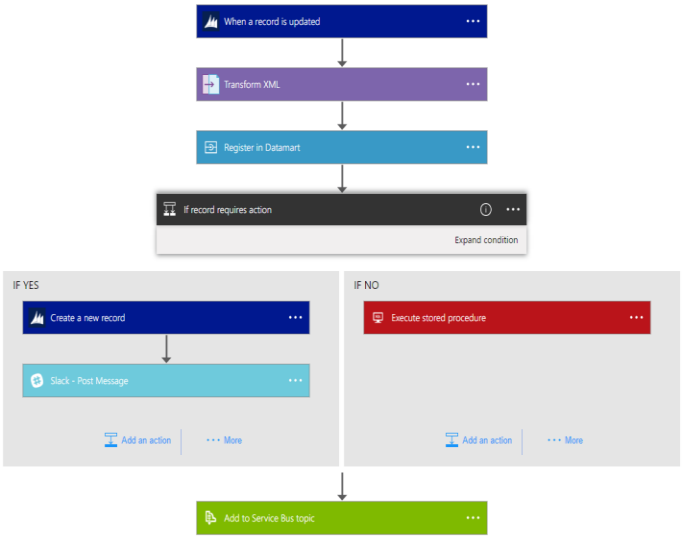
As easy as connecting to cloud services



Introducing Azure Logic Apps

Powerful Integration.
Hyperscale Workflow Engine, born in Azure

- Faster integration using innovative Visual Designer
- Easy workflow creation with triggers and actions
- Mashup applications, data and services
- Built for mission critical Enterprise Integration
- Create, deploy, manage and monitor



Logic Apps

Cloud APIs and platform functionality

Over 120 built-in connectors

Hosted and managed within the platform

Scales to meet your needs

First class designer experience

Rapid development

API connections

Authenticate once and reuse

Differentiate connection configuration

Simple to deploy

Portal experience for managing API Connections

SaaS

- 10to8
- Act!
- appFigures
- Asana
- Azure Active Directory
- Azure API Management
- Azure App Services
- Azure Automation
- Azure Cognitive Face API
- Azure Cognitive LUIS
- Azure Cognitive Text Analytics
- Azure Cognitive Vision
- Azure Data Lake Store
- Azure Document DB
- Azure Event Hubs
- Azure Functions
- Azure Machine Learning
- Azure Resource Manager
- Azure Service Bus
- Azure SQL
- Azure Storage Blob
- Azure Storage Queues
- Basecamp2&3
- Benchmark Email
- Bing Search
- BitBucket
- Bitly
- Blogger
- Box
- Buffer
- Campfire
- CapsuleCRM
- Chatter
- Common Data Service
- Disqus
- DocuSign
- Dropbox
- Dynamics AX Online
- Dynamics CRM Online
- Dynamics CRM Service Bus
- Dynamics Financials
- Dynamics Operations
- Easy Redmine
- Eventbrite
- Facebook
- FreshBooks
- Freshdesk
- GitHub
- Gmail
- Google Calendar
- Google Contacts
- Google Drive
- Google Sheets
- Google Tasks
- GoTo Meeting
- GoTo Training
- GoTo Webinar
- Harvest
- HelloSign
- HipChat
- Infusionsoft
- Inoreader
- Insightly
- Instagram
- Instapaper
- Intercom
- JIRA
- LeanKit
- LiveChat
- MailChimp
- Mandrill
- Medium
- Microsoft Translator
- MSN Weather
- Muhimbi PDF
- Nexmo
- Office 365
- Office 365 Users
- Office 365 Video
- OneDrive
- OneDrive for Business
- OneNote
- Outlook.com
- Outlook Customer Manager
- Outlook Tasks
- PagerDuty
- Paylocity
- Pinterest
- Pipedrive
- Pivotal Tracker
- Power BI
- Project Online
- Redmine
- Salesforce
- Salesforce Chatter
- SendGrid
- SharePoint Online
- Slack
- SmartSheet
- SparkPost
- Stripe
- Survey Monkey
- Teradata
- Todoist
- Toodledo
- Trello
- Twilio
- Twitter
- Typeform
- UserVoice
- Vimeo
- VS Team Services
- Webmerge
- Wordpress
- Wunderlist
- Yammer
- YouTube
- Zendesk

Protocols/Native

- HTTP, HTTPS
- HTTP Webhook
- FTP, SFTP
- SMTP
- RSS
- Compose, Parse JSON
- Query, Join, Table, Select
- Schedule, Wait
- Terminate
- Workflow

XML & EDI

- XML Validation
- Transform XML (+Mapper)
- Flat File Encode
- Flat File Decode
- X12
- EDIFACT
- AS2
- Integration Account Artifact Lookup

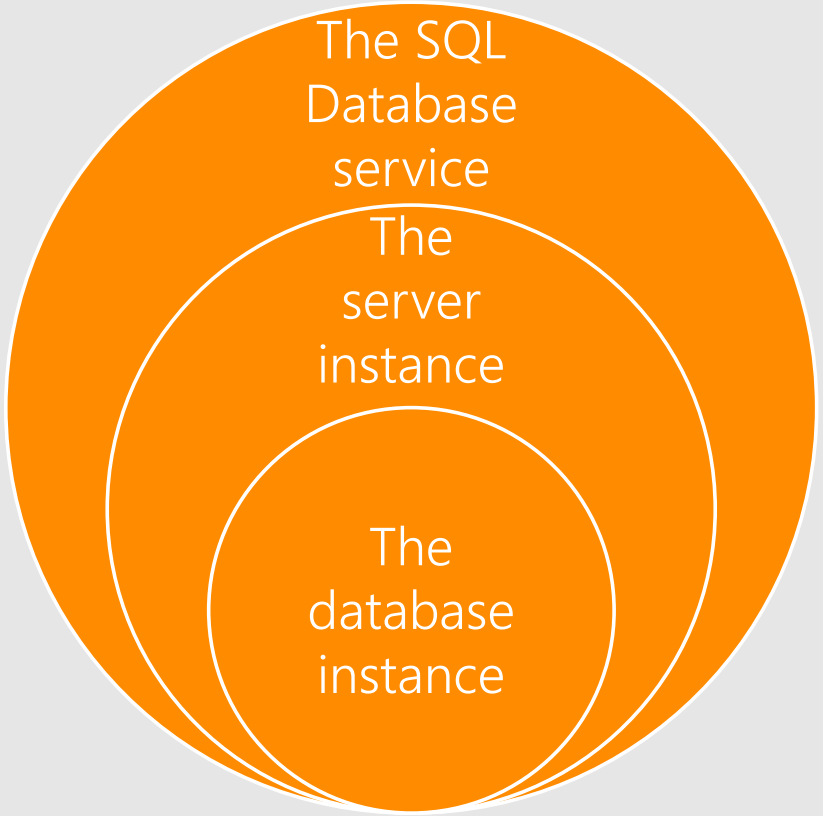
Hybrid

- BizTalk Server
- File System
- IBM DB2
- Informix
- Oracle DB
- SharePoint Server
- SQL Server
- SAP
- Websphere MQ

Demo

- Creating an Azure SQL Database Instance
- Demonstration: Creating a SQL Database

Creating an Azure SQL Database Instance



Demonstration: Creating a SQL Database

In this demonstration, you will learn how to:

- Create a SQL Database instance by using the Portal
- Create a logical server instance by using the Portal

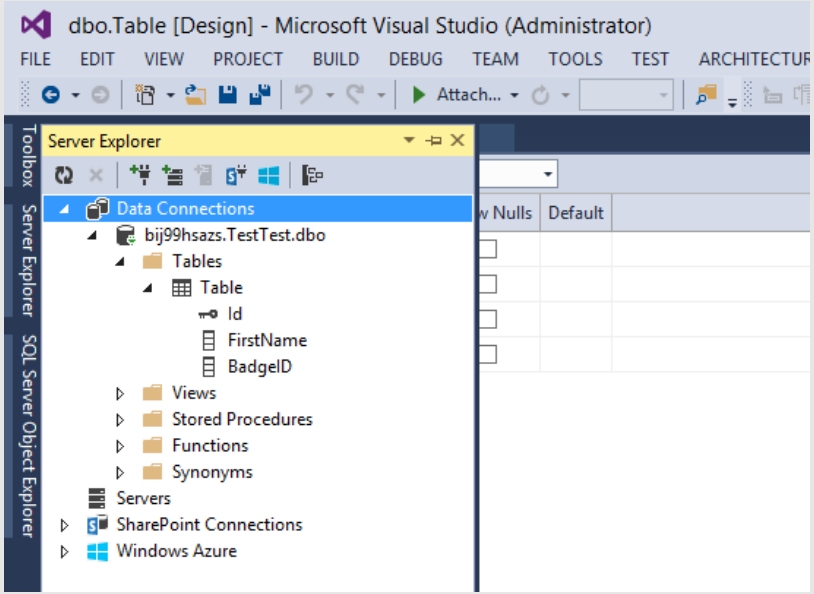
SQL Server Management Studio

- You can connect to an Azure SQL Database using many of the tools that you use right now such as SQL Server Management Studio:
  - First, you must add your IP address to the firewall rules of allowed IP addresses. This is done in the configuration page for the server
  - The Server Name is viewable on the dashboard of the database or the server
  - Use SQL Server Authentication and the username/password you set up in the creation
    of the server

Visual Studio Server Explorer

You can use Server Explorer to manage SQL databases in the same way as you use SQL Server Management Studio

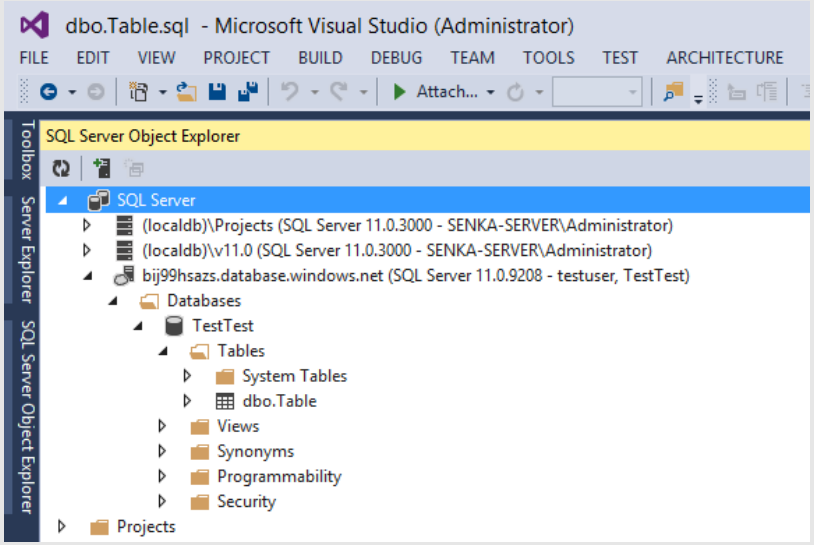
The improved query editor and updated preview tools can be helpful for determining the collateral of changes made through a SQL script



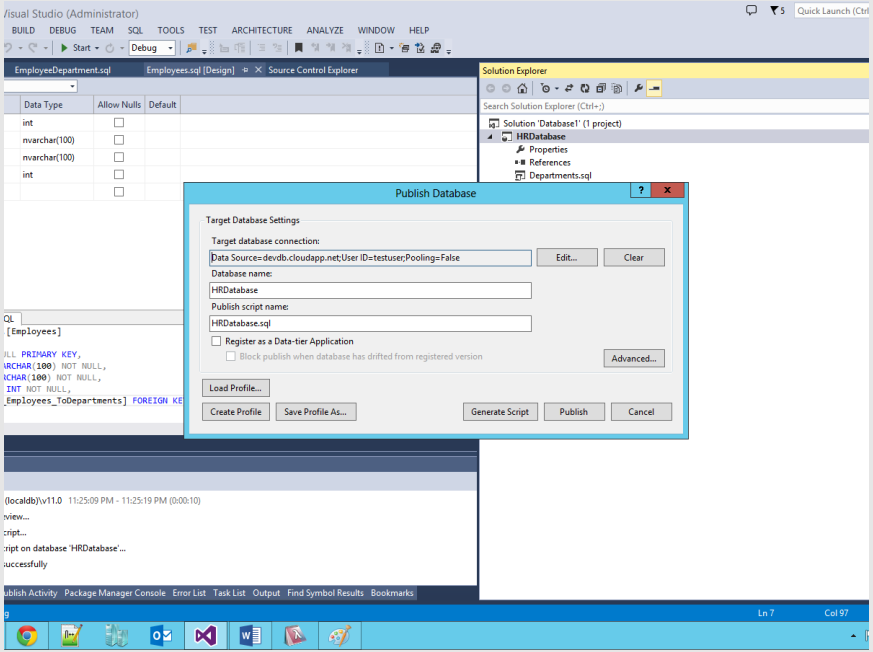
Visual Studio SQL Server Object Explorer

You can use the SQL Server Object Explorer to manage your local databases or SQL databases in Azure

You can access the SQL Server Object Explorer by right-clicking a database in Server Explorer



Visual Studio Database Projects



Database projects in Visual Studio allow you to:

- Place the schema of a database in-development within source control
- Publish the database to a SQL Database instance

Migration Tools

- You can use SQL Server Integration Services to define a migration plan for on-premises databases
- SQL Database Migration Wizard analyzes your existing database and generates a script (and bulk copy files) to migrate your database
- Azure Websites Migration Assistant uses SQL Management Objects to analyze your existing database and migrate it

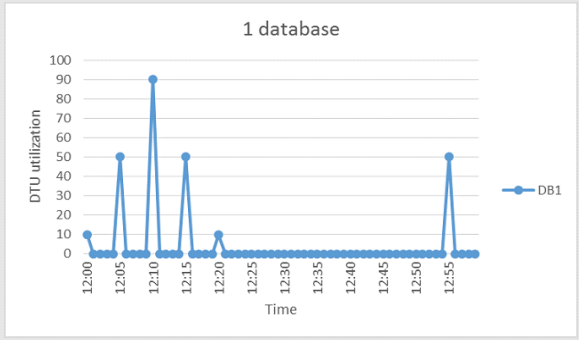
Elastic Database Pools

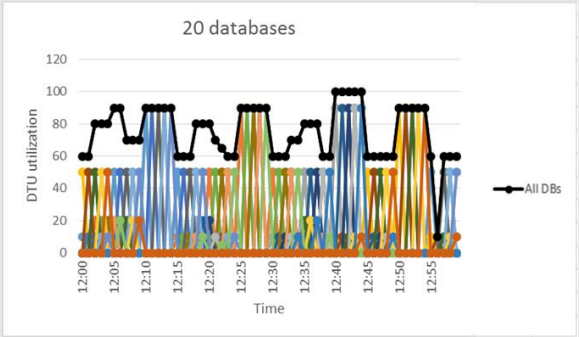
- Collection of Elastic Database Transaction Units (eDTUs) and storage resources
- Deploy several databases in the same Elastic pool
- Resources are shared between all of the databases within the pool
- Cost-effective solution to manage multiple databases and implement sharing patterns

When to Use Elastic Pools

Databases with specific utilization patterns:

- Low average utilization
- Infrequent usage spikes
- Large differences between peak usage per database
- Peak usage of databases occurs in different point in time





- Recovery Options for Azure SQL Database
- Azure SQL Databases Geo-Replication

Recovery Options for Azure SQL Database

- You can count on infrastructure redundancy in the Azure data centers for the most basic scenarios
- SQL Database also provides its own set of options for HADR:
  - Built-in replicas:
    - Transactions are not considered committed to the database until they are written to the target DB, one primary replica, and two secondary replicas
  - Backup and restore:
    - Allows you to protect against errant transactions
    - Database is backed up as a whole and can be recovered through the portal
    - The retention period (in days) of your backup is based on your selected service tier

High Availability for Azure SQL Databases

- Database copy:
  - You can create a one-time copy of the database within the same region
  - The copy is disconnected from the source database so any future transactions will not be applied to the copy
- The import and export service:
  - You can import and export BACPACK files from a database instance either automatically or manually
  - You can use this service to offer a minimum HADR option for web and business databases that do not support the point-in-time recovery option
  - You also can use this service for custom backup and recovery scenarios

Azure SQL Databases Geo-Replication

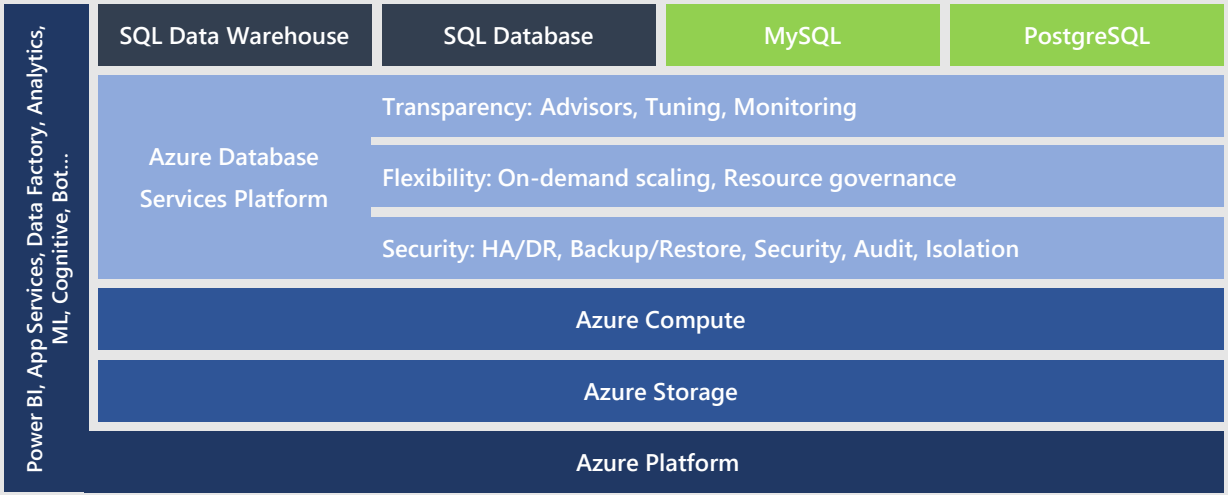
- Active geo-replication is available for Premium SQL Database instances
  - This feature is asynchronous by default and guarantees that replicas will be eventually consistent
  - You can replicate transactions to as many as four copies of the database
  - Replicas can exist in different regions for geo-redundancy
- You can use the replica of the database as a read-only data source in load-balancing scenarios
  - Example: An application uses the primary database for line-of-business functionality and the replica for reports

Additional Managed Database Services

- Azure Database for MySQL
- Azure Database for PostgreSQL

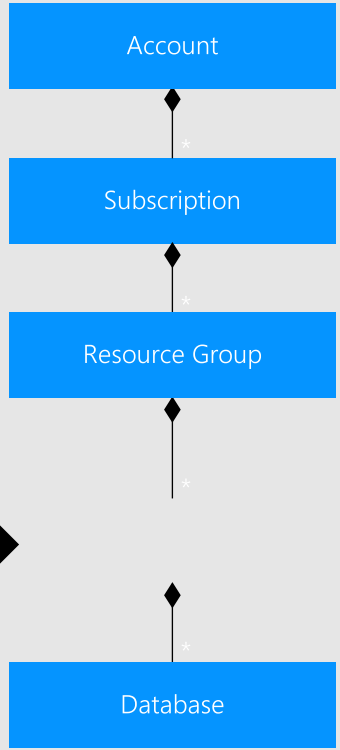
Managed Database Platform

Unified platform for all managed database services

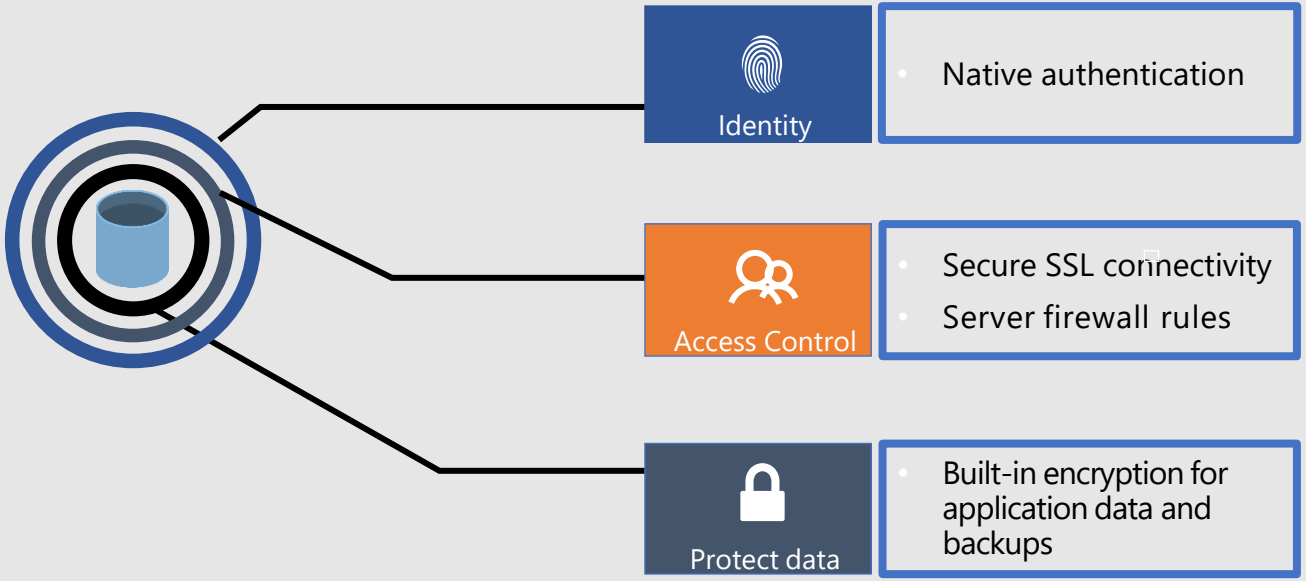


Conceptual Model

- Logical with provisioned compute and storage resources
- Connection endpoint for PostgreSQL/MySQL server
- Can create one to many user databases
- Pinned to a region
- Policy scope, e.g., firewall rules, recovery, monitoring, and management



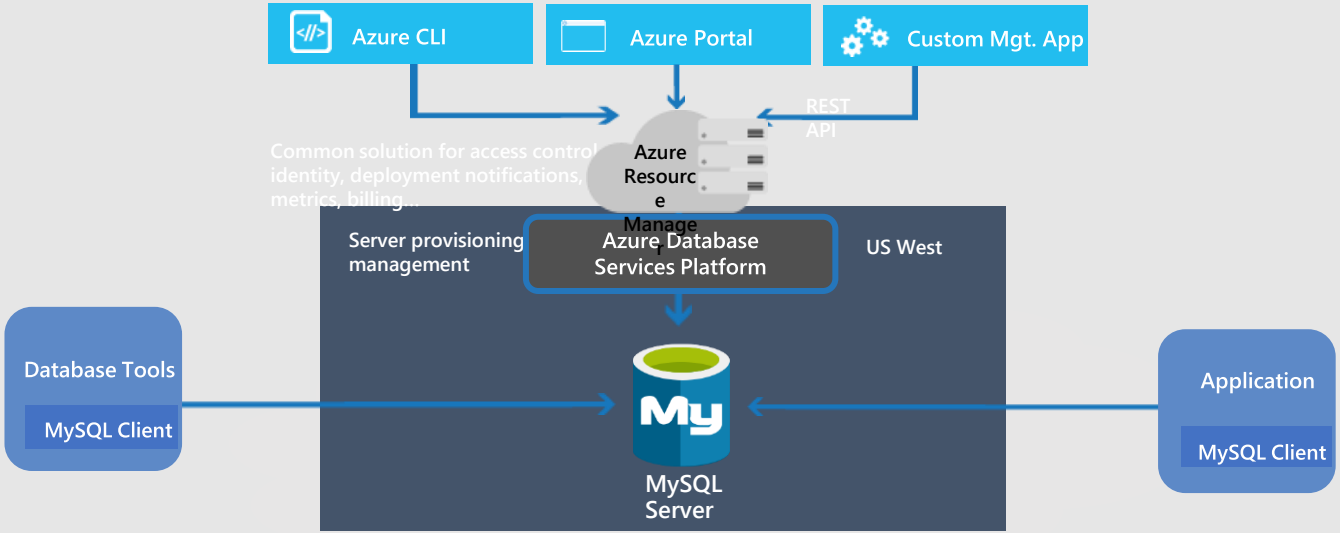
Integrated Security



Azure Database for MySQL

- Based on MySQL Community Edition
- Shares common features with other managed database services:
  - Provision in minutes with built-in high availability
  - Predictable performance
  - Scale on the fly without application downtime
  - Secured to protect sensitive data at-rest and in-motion
  - Automatic backups and Point-In-Time-Restore for up to 35 days
  - Deep integration with Azure Web Apps

Connecting to MySQL Instance



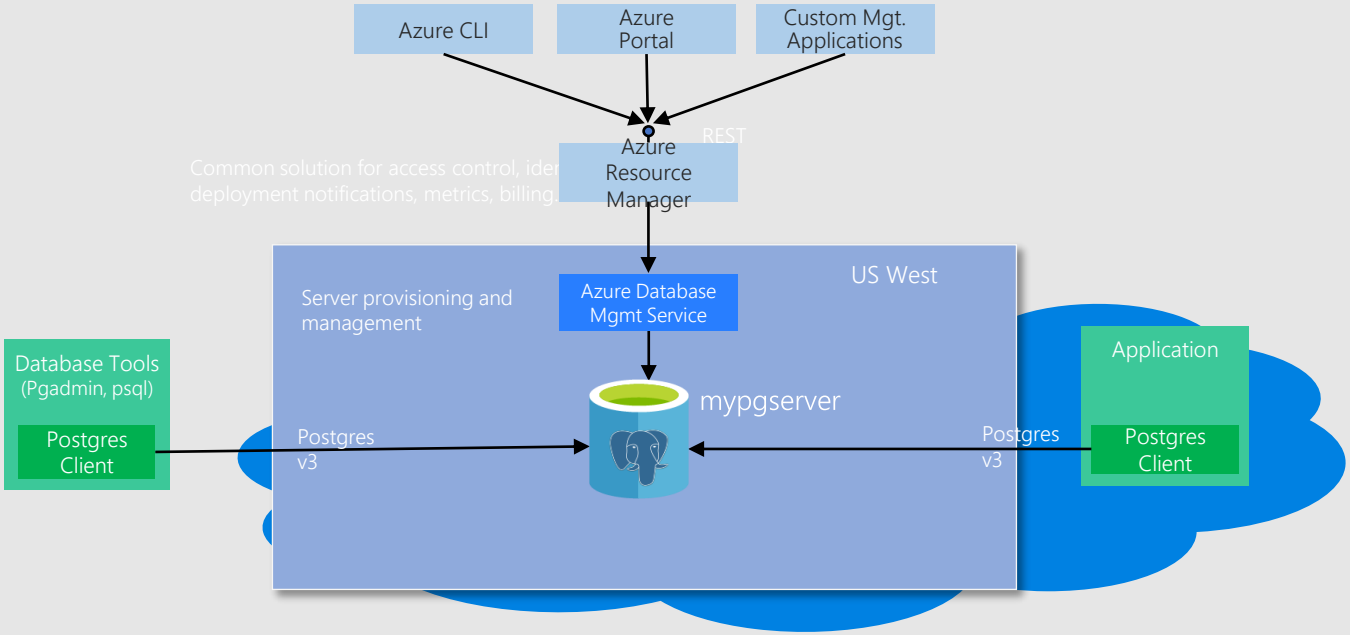
Integration with Azure Web Apps

- Tightly integrated with Azure Web Apps as a DB Provider
- Deploy a new Web App + MySQL and add your own app
- Preconfigured Web Apps:
  - PHP
  - Wordpress
  - Drupal
  - Ghost
- Deploy through CLI using your own customized ARM deployment template

Azure Database for PostgreSQL

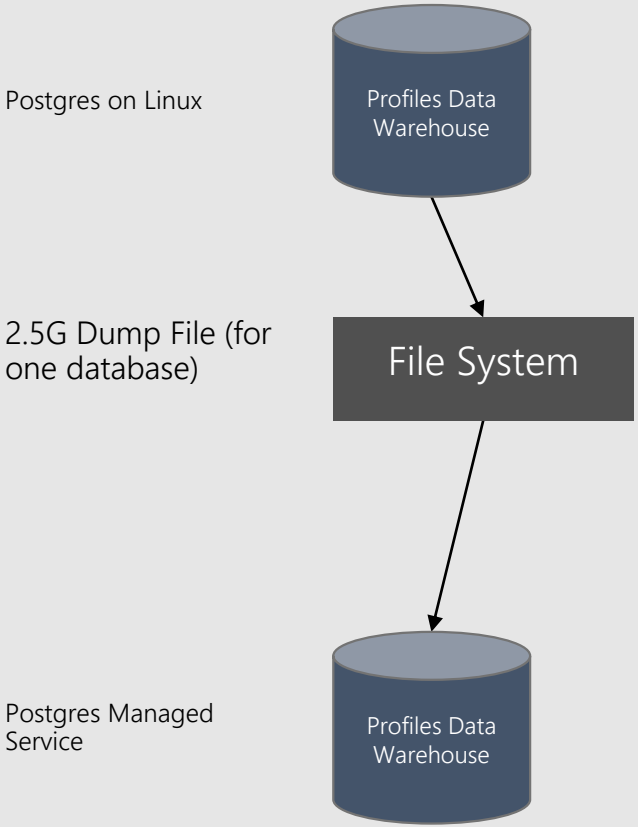
- Based on PostgreSQL Community Edition
- Shares common features with other managed database services:
  - Provision in minutes with built-in high availability
  - Predictable performance
  - Scale on the fly without application downtime
  - Secured to protect sensitive data at-rest and in-motion out-of-the-box
  - Automatic backups and storage for recovery to any point up to 35 days
  - Use native tools, drivers and libraries

Connecting to PostgreSQL Instance



Migrating to Managed PostgreSQL

- Create service using portal
- Export an existing database using pg_dump
- Create database and users on PostgreSQL service
- Import using psql



Lab: Storing Event Data in Azure SQL Databases

- Exercise 1: Creating an Azure SQL Databases Instance
- Exercise 2: Using Entity Framework with Local SQL Server
- Exercise 3: Using Entity Framework with Azure SQL Databases

Lab Scenario

Now that the web application is ready for publishing, you can begin migrating the web application to Azure by creating a database in Azure. You decided to use a database initializer and Entity Framework Code First to automate the creation of your database in SQL Database.

Azure Data Studio