Cloud Powered Mobile Apps with Microsoft Azure

Malte Lantin

Technical Evanglist

Microsoft Azure

Malte Lantin

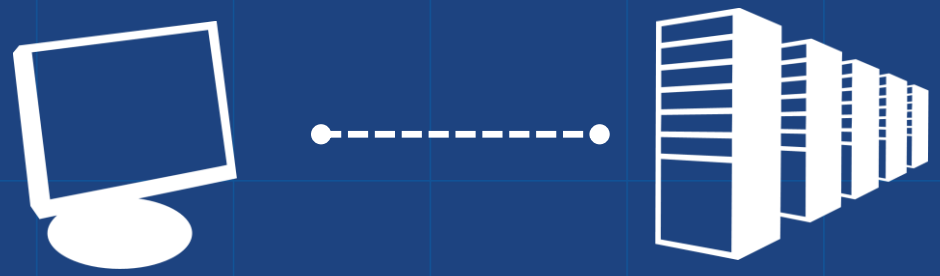
Technical Evangelist, Microsoft Deutschland

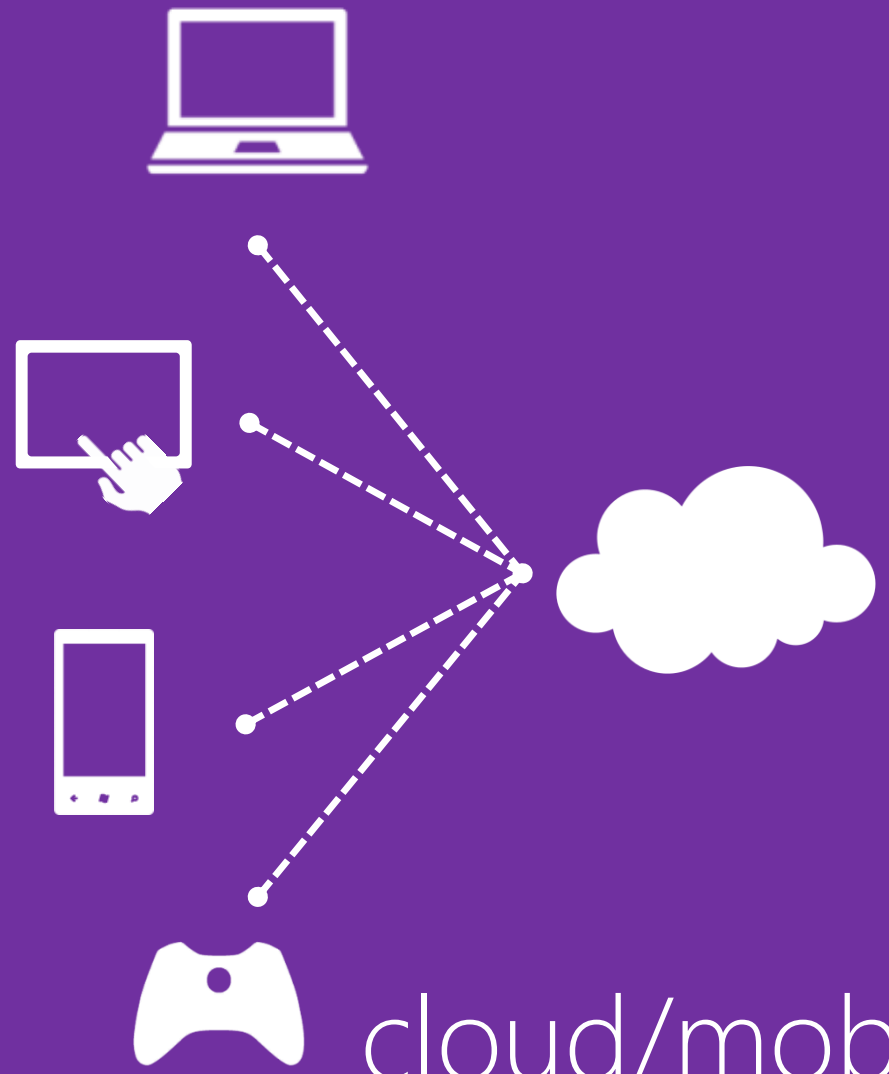
- Fokus auf Microsoft Azure, App-Entwicklung
- Student Partner Program Lead
- Blog unter http://blogs.msdn.com/b/malte_lantin
- Twitter @MalteLantin
- Malte.Lantin@Microsoft.com

Seit 5 Jahren bei Microsoft

- Microsoft Student Partner / Universität Duisburg-Essen
- Diplom Wirtschaftsinformatiker
- Seit Anfang 2010 bei Microsoft in unterschiedlichen Rollen (Product / Audience Marketing, Technical Evangelism)
- Sprecher auf diversen Konferenzen und Veranstaltungen

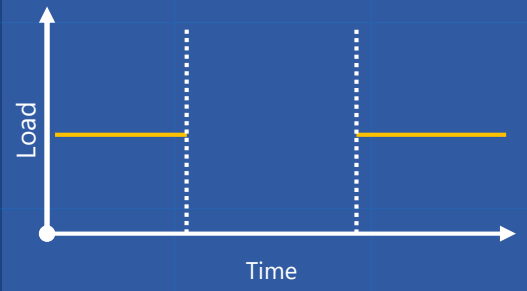
server/client



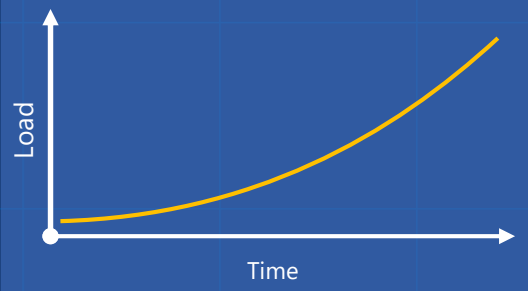


cloud/mobile

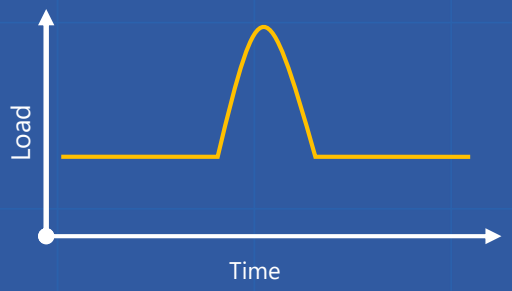
Scenarios for the cloud



Temporary Resources



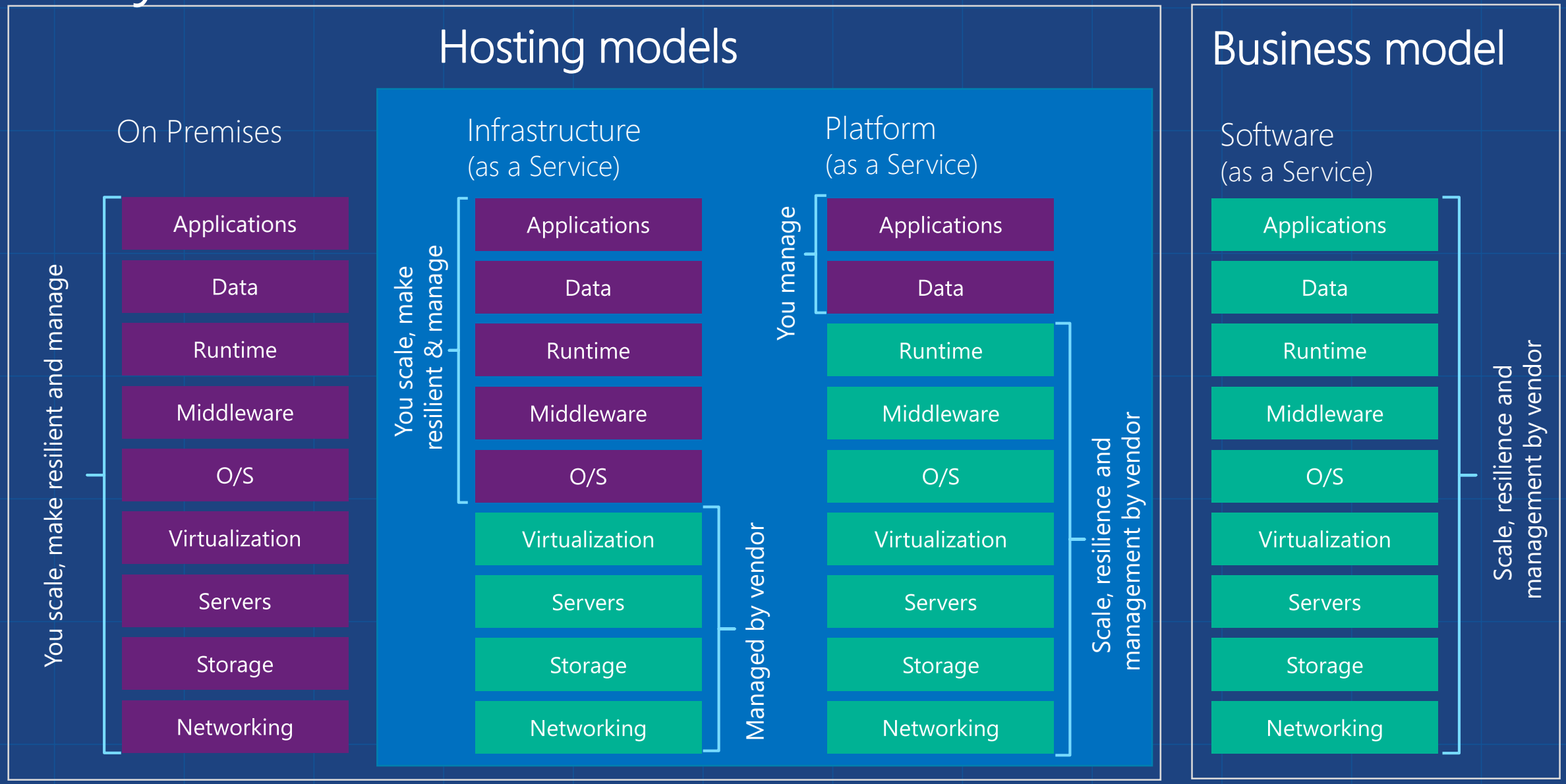
Flexible Growth



Load Peaks

Access from anywhere and any time

Why the cloud?



Azure

Microsoft Azure Services

How to use Microsoft Azure

Management portal

Pure HTML / JavaScript – works in every modern browser

Scripting
(Windows, Linux and Mac)

CLI for Windows, Linux & Mac – PowerShell for Windows

REST API

Use in you own applications – used by Management Tools

Web Services und Microsoft Azure

- REST-based Web Services are an essential part of the platform

1. Provisioning of Web Services as components of larger cloud services and modern web applications
   - e.g. Azure Mobile Services, API-Management
2. Management of services in Microsoft Azure through web services
   - z.B. POST https://management.core.windows.net/<subscription-id>/services/hostedservices/<cloudservice-name>/deployments

Where do we come from?

- Remote Procedure Calls (RPC)

- Actions in endpoints
  - GetCustomer
  - InsertInvoice
  - DeleteCustomer

- Every endpoint corresponds to one URI

Representational State Transfer (REST)

- REST is not a standard but uses standards
  - Architectural Concept
- Roy Fielding in 2000
  - "Architectural Styles and the Design of Network-based Software Architectures"
- HTTP Stack, explicit use of HTTP methods / verbs

Design principles of REST

- stateless
- URIs with folder structure
- XML, JavaScript Object Notation (JSON) or both
  - Transfer of data snapshots in the message body
  - Not commands or methods in message body

Anatomy of HTTP-messages

Request
- Request Line
- Request Headers
- Request Body

Antwort Response
- Status Line
- Response Headers
- Response Body

Example: GET

- Request
  - GET http://localhost:21697/api/employees/12345 HTTP/1.1
  - User-Agent: Fiddler
  - Host: localhost:21697

- Response
  - HTTP/1.1 200 OK
  - Content-Type: application/json; charset=utf-8
  - Date: Tue, 04 Nov 2014 15:48:03 GMT
  - Content-Length: 50

  - {"Id":12345,"FirstName":"John","LastName":"Human"}

HTTP-Methods

- Create, Read, Update und Delete (CRUD) mapped to http methods
  - Create → POST
  - Read → GET
  - Update → PUT
  - Delete → DELETE

- Design principles: nouns instead of verbs

Example: POST

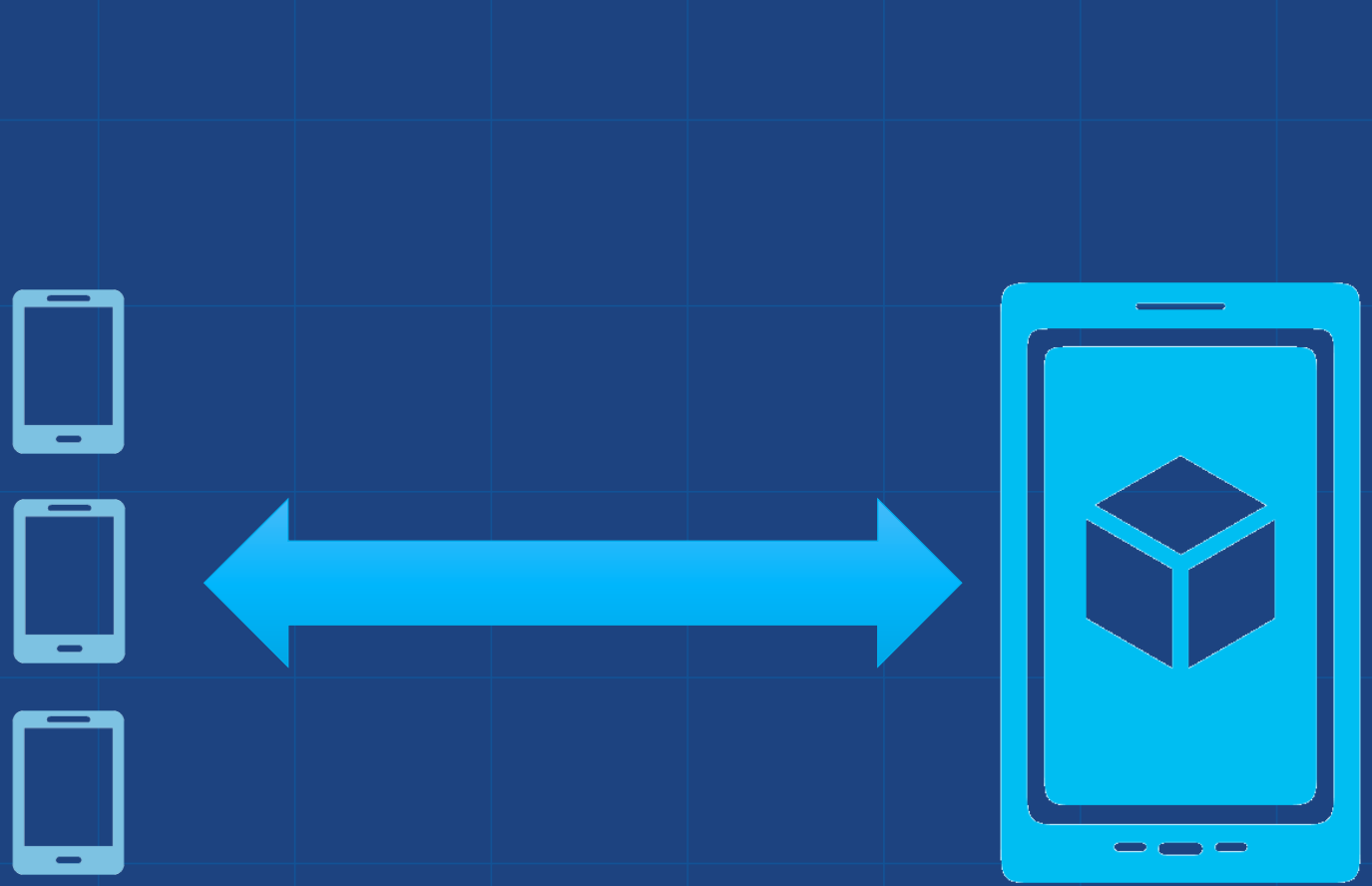
https://management.core.windows.net/<subscription-id>/services/hostedservices/<cloudservice-name>/deployments

http://msdn.microsoft.com/en-us/library/azure/jj157194.aspx

Azure Mobile Services

- Mobile Services
- Features and Demos
- Advanced Features
- Scaling and Pricing

What are Mobile Services?



Storage

Authentication

Logic

Push

Scheduler

Demo: Getting Started

Your first Mobile Service

Structured Storage

- Powered by SQL Database
- Supports rich querying capabilities
- Dynamic Schematization
- Data management in:
  - Azure Portal
  - SQL Portal (Silverlight)
  - SQL Management Studio
  - REST API
  - Azure CLI Tools
  - SQL CLI

The REST API

Base Endpoint: **https://MobileService.azure-mobile.net/tables/***

Action	HTTP Method	URL Suffix
Create	POST	/TodoItem
Read	GET	/TodoItem?filter=id%3D42
Update	PATCH	/TodoItem/id
Delete	DELETE	/TodoItem/id

JSON to SQL Type Mappings

JSON Type	T-SQL Type
Numeric values (integer, decimal, floating point)	Float(53)
Boolean	bit
DateTime	DateTimeOffset(3)
String	Nvarchar(max)

Auto-generated Columns

- id – unique guid
- __createdAt – date
- __updatedAt – date
- __version – timestamp
  - Helps with concurrency and offline

Backend Logic:
JavaScript & .NET

Server Side Table Scripts

Node.js scripts

.NET Web API backend in Visual Studio

Passes through to SQL by default

MongoDB, Table Storage, SQL out of the box

Intercept CRUD requests to tables

Fully customizable

.NET Backend

- Web API based w/ additional functionality, developed in and deployed from Visual Studio
- TableController data context can map to SQL, Table Storage, Mongo, etc
- Pull in NuGet modules and other .NET libraries
- Set permissions with attributes on classes
- Local Debug

Custom API

- Non-table based endpoints
- Accessible from
  - GET
  - POST
  - PUT
  - PATCH
  - DELETE
- For node.js logic in scripts like table endpoints
- For .NET delivered through a WebAPI
- Expose any functionality you want

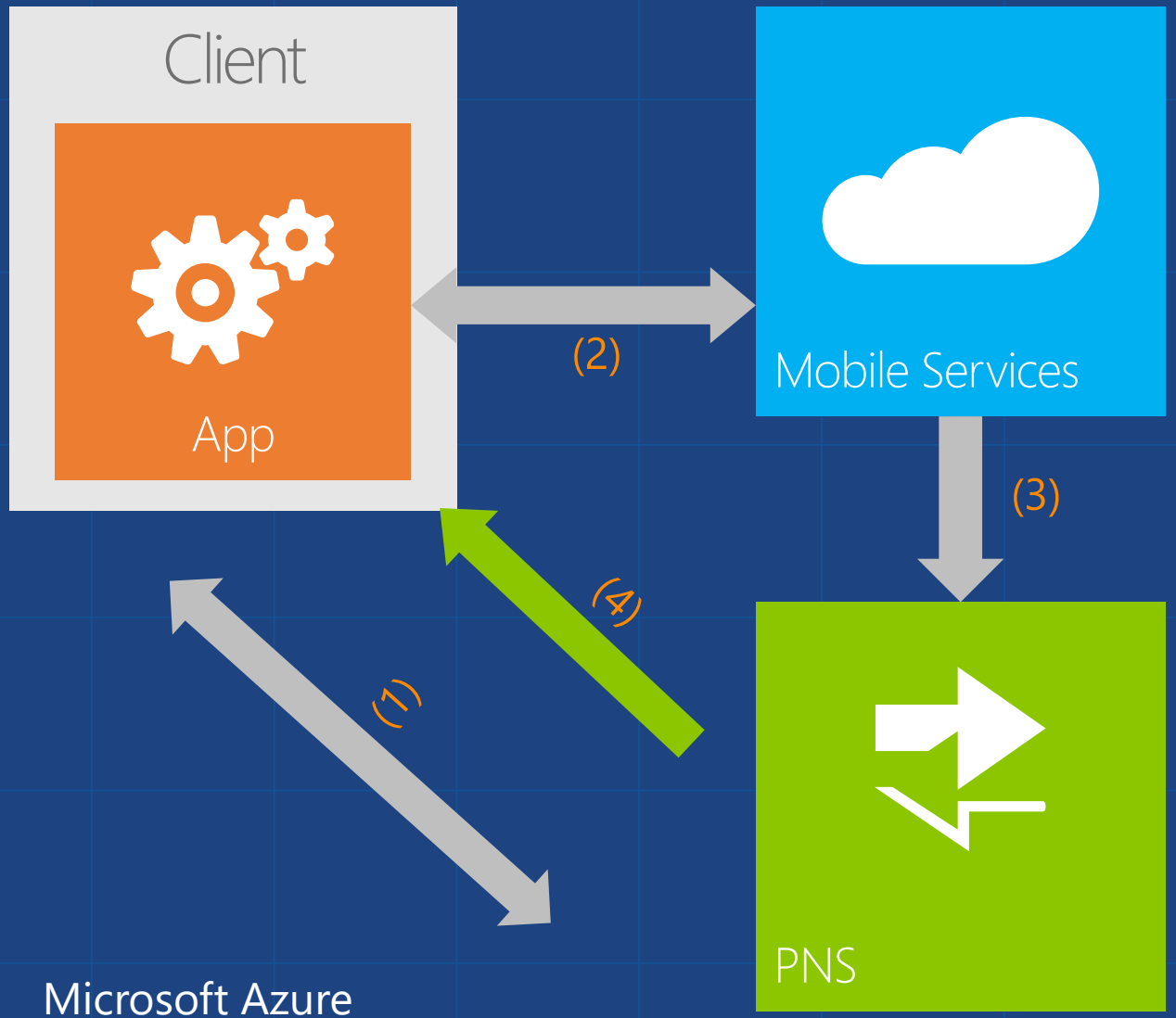
Demo: Dynamic Schema

File Storage

- Don’t store files in Mobile Services DB
- Use Mobile Service to provide a Valet Key Pattern to Blob Storage, aka BLOB SAS
  - Create table / custom API script to talk to Blob Storage
  - Script generates Shared Access Signature (SAS) URL
  - Script returns SAS URL to client app
  - Client app uploads data to blob storage directly
  - Store file URL in Mobile Service DB as needed
- Blob storage costs less and is built for redundant file storage
- Resource Broker extension for .NET and Node backends

Push Notifications

Push Notification Flow



1. Register for push notifications with PNS
2. Send your identifier to Mobile Service
3. Send push from server scripts
4. PNS delivers notification to device

Notification Hubs

- Separate from Mobile Services
  - Can be used regardless of whether you're storing data in Azure
- Extremely scalable push notifications
- Cross platform support
  - Push to iOS, Android, Kindle, Windows Phone, Windows Store
- Tags (i.e. tie my registration to this topic or user ID)
- Templates (i.e. when I get a push, send it in this format)
- Server SDKs for .NET, Java, and Node (also open as REST API)

Authorization & Authentication

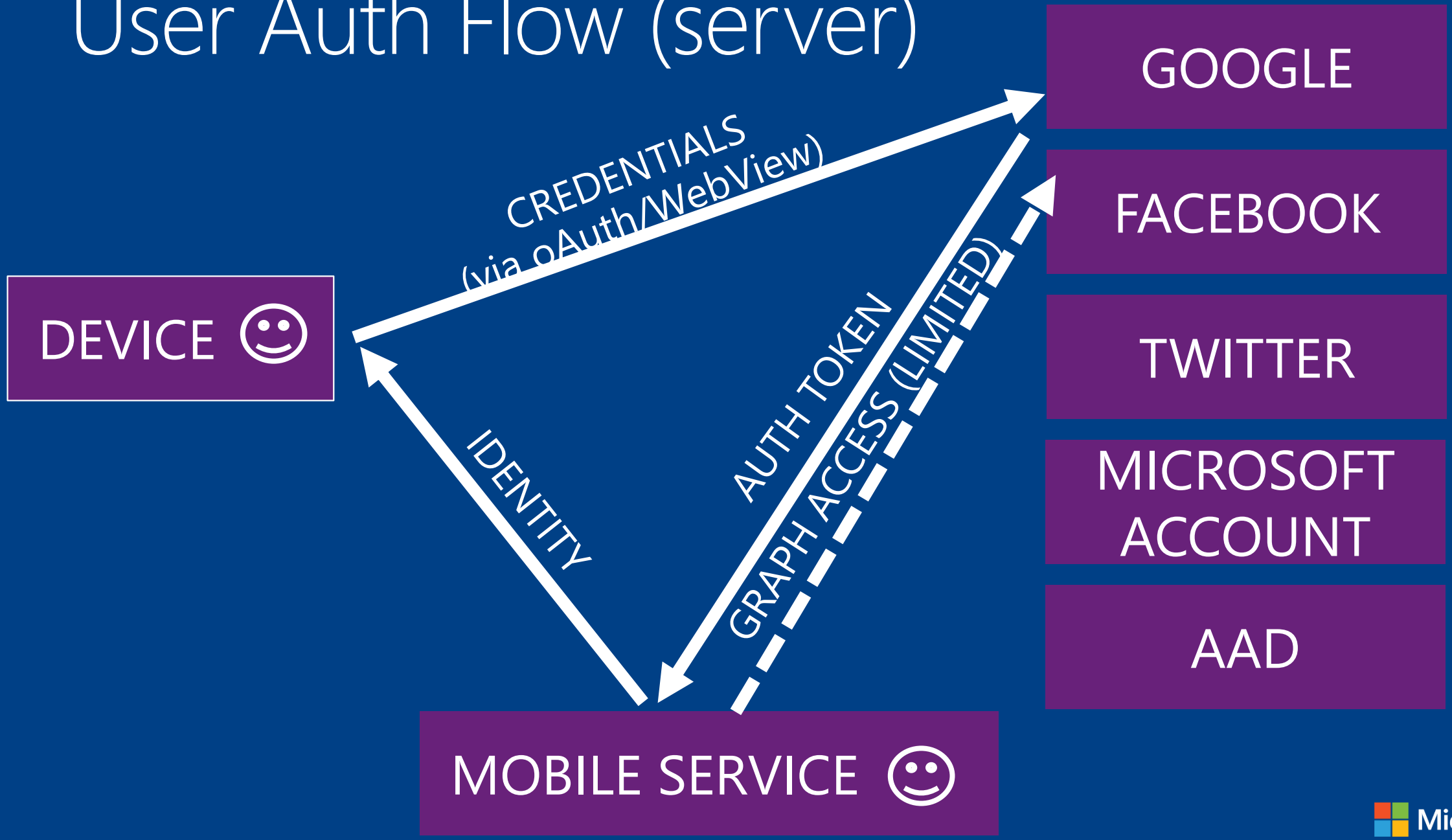
Data Authorization

Per HTTP method auth options:

- App Key Required
  - Not ideal for production use
- Everyone
- Authenticated Users
- Admins and other scripts
  - Requires Master Key as header

401 Unauthorized response if security check fails

User Auth Flow (server)



The User object

User.level

- Admin
- Authenticated
- Anonymous

User.userId

- Provider:id or undefined

User.getIdentities()

- UserId
- Provider Access Token / Secret
- Basic user information (i.e. name, username, locale, picture, link)

Script Source Control

Script Source Control

- Creates a Git repo in Azure
- Node.js only, turned on by default
- Access table, scheduler, custom API, shared scripts and permissions
- Pushing changes triggers a redeploy of your Mobile Service
- Enables installing NPM modules (for Node backends)

- Shared Scripts
  - Create scripts that can be used from all other scripts in your Service

Diagnostics
Logging
Scale

Diagnostics, Logging, Scale

API Calls, # of Devices, Data Out

Console Logging (auto error logging)

Scale Service Based off API Calls

Auto-scale to Save Money

Free Tier for Mobile

Free Tier for SQL

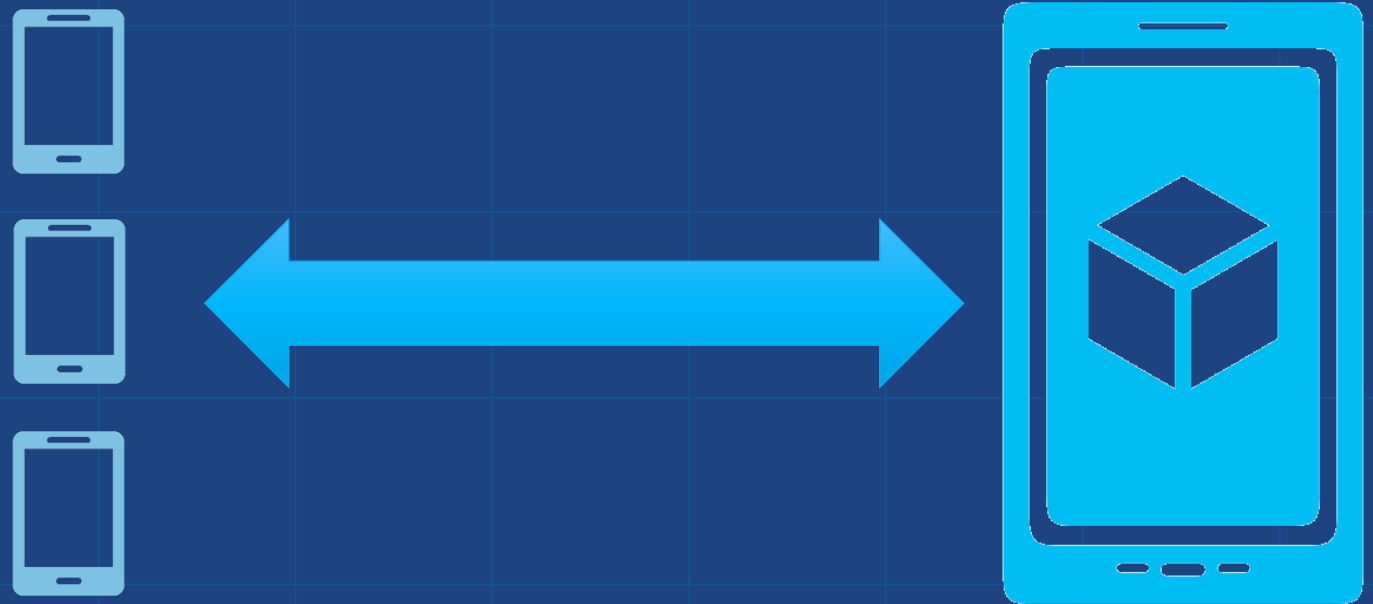
Mobile Service Scaling

- Free: 500k calls / month / subscription
- Basic: 1.5M calls / unit (6) / month
- Standard: 15M calls / unit (10) / month

Demo: Scaling

Leaving Free Mode and Turning on AutoScale

Azure Mobile Services



Storage

Authentication

Logic

Push

Scheduler

Task for next week

Use Azure Mobile Services to add user authentication to your apps

- http://aka.ms/azuremobileauth
- Choose MS Account, Facebook, Twitter, Google

Save information per user after login