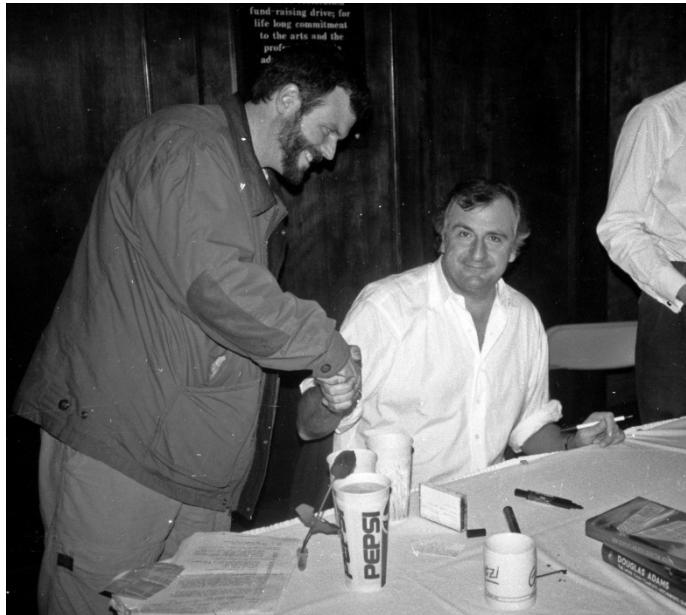


# A Hitchhiker's Guide to Azure Mobile Apps



# David Giard

Microsoft Technical Evangelist

- @DavidGiard
- DavidGiard.com
- TechnologyAndFriends
- dGiard@microsoft.com



@DavidGiard

This presentation is dedicated to Bill Fink



@DavidGiard

# This presentation is dedicated to Bill Fink



@DavidGiard

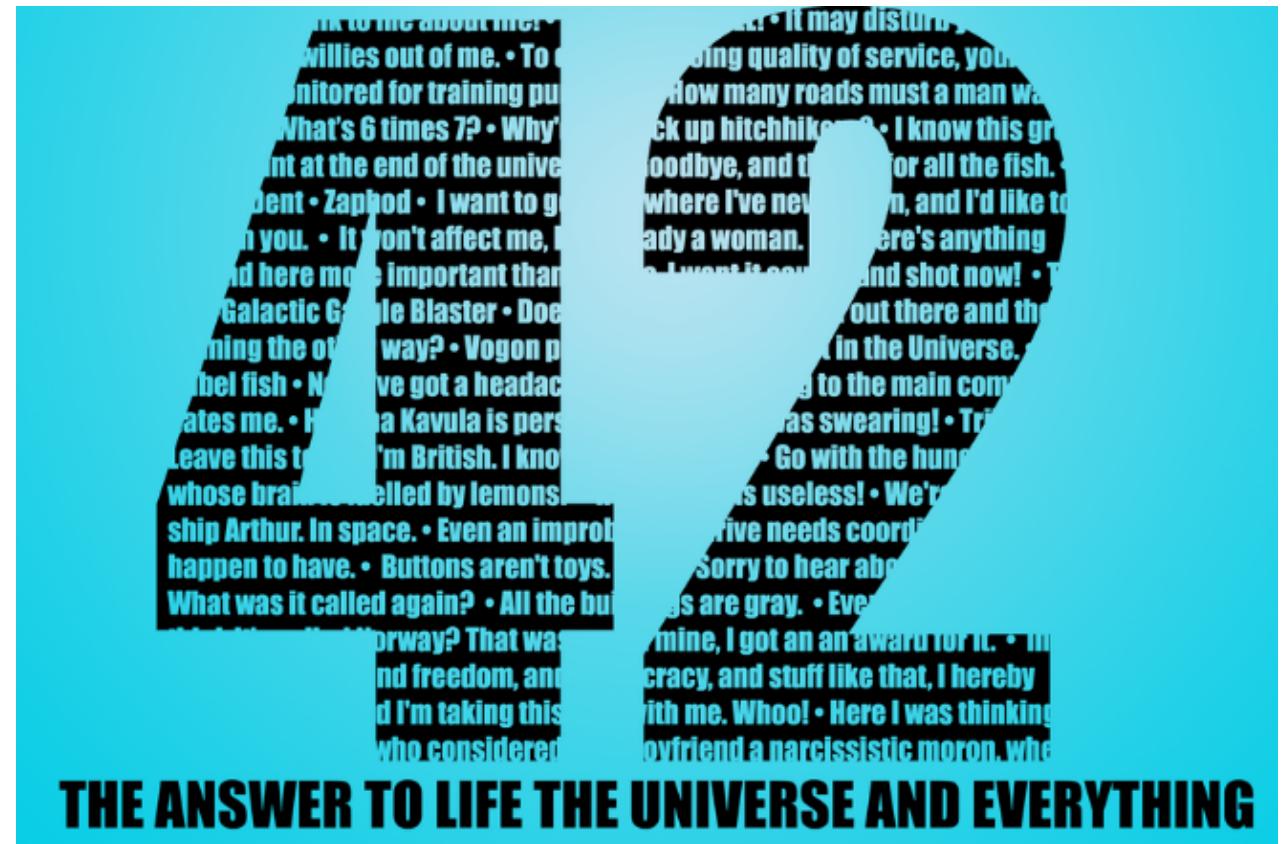
# Agenda

- Why Azure Mobile Apps?
- Mobile Architecture
- Common Data Access
- Customization
- Dynamic Data
- Client Code
- Identity and Permissions
- Scaling
- API Scripts
- Notifications



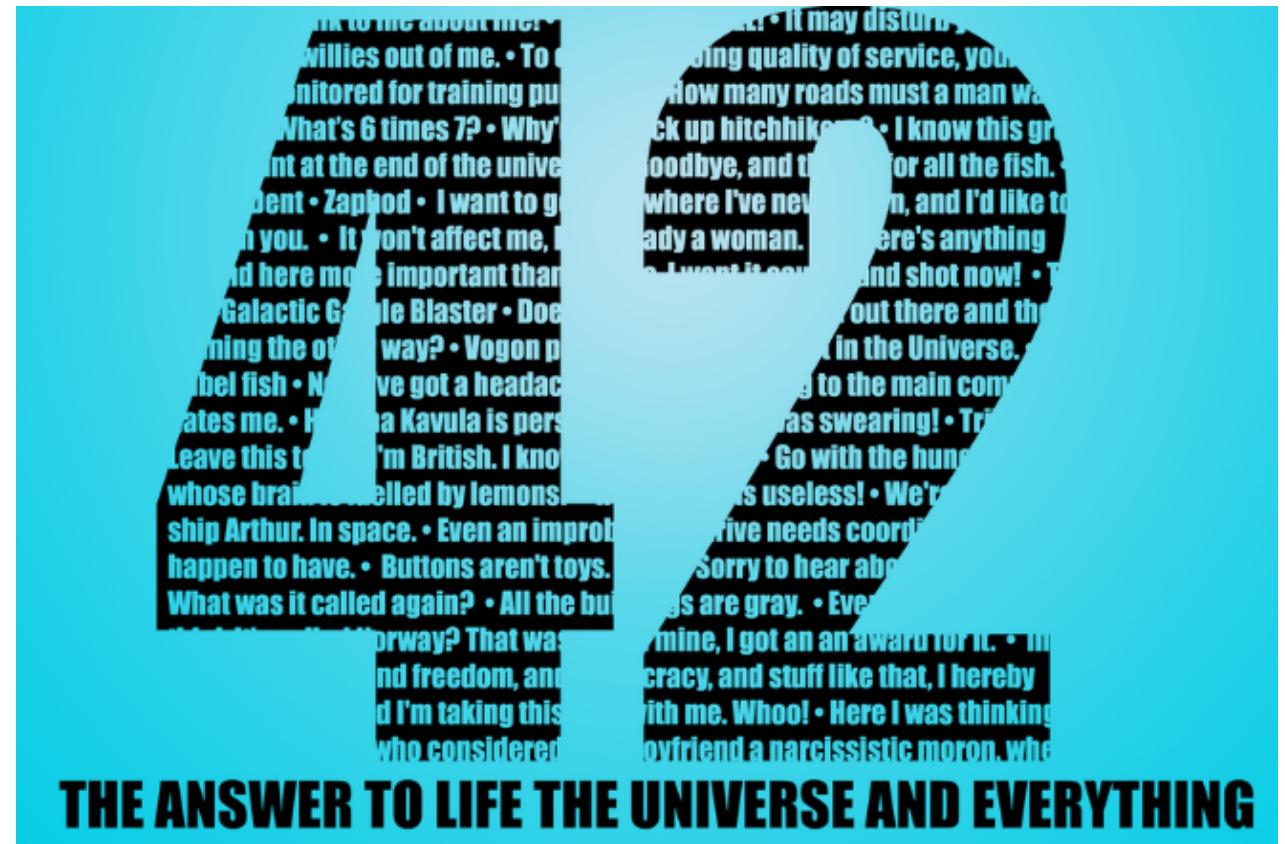
# Why Azure Mobile Apps?

- Integrate with
  - Web Apps
  - Logic Apps
  - API Apps
- All features of Web Apps

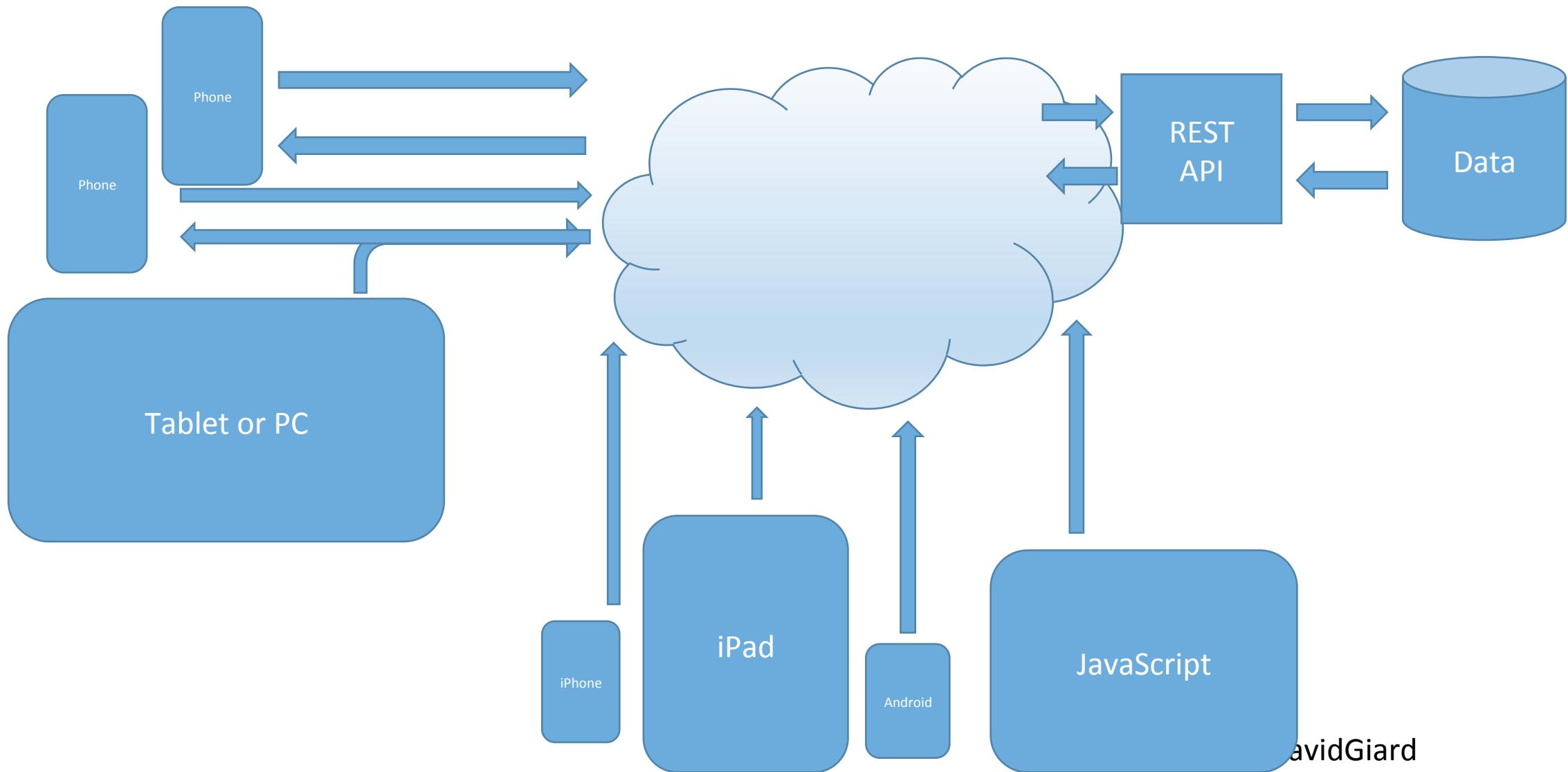


# Why Azure Mobile Services?

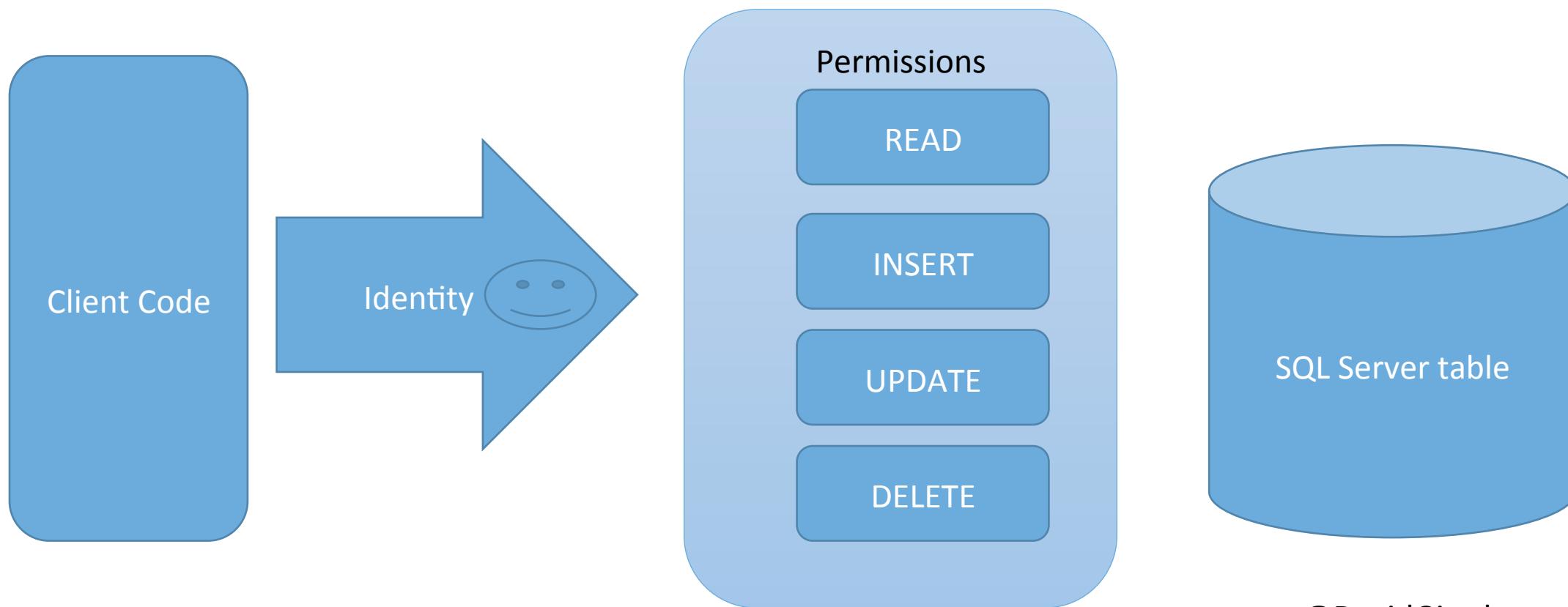
- Frees you from plumbing code
- Handles API Changes
- Cross-Platform solution
- Choose JavaScript or .NET



# Mobile Architecture



# Pieces of Azure Mobile Apps



# Creating a Mobile App

The screenshot shows the Microsoft Azure portal interface for creating a new mobile application.

**Left Panel (New):**

- Search bar: Search the marketplace
- Marketplace categories:
  - Compute
  - Web + Mobile** (selected)
  - Data + Storage
  - Data + Analytics
  - Internet of Things
  - Networking
  - Media + CDN
  - Hybrid Integration
  - Security + Identity
  - Developer Services
  - Management
  - Container Apps
- Recent items:
  - Mobile App (Microsoft)
  - SQL Database (Microsoft)

**Middle Panel (Web + Mobile):**

FEATURED APPS

- Web App**: Enjoy secure and flexible development, deployment, and scaling options for your web app.
- Mobile App**: A scalable and secure backend that can be used to power apps on any platform – iOS, Android, Windows or
- API App**: Scalable RESTful API with enterprise grade security, simple access control and auto SDK generation
- Logic App (preview)**: Automate the access and use of data across clouds without writing code
- App Service Environment**: Deploy Azure App Services in your Virtual Network with greater scaling options
- API Management**: Ensure a successful API program through developer engagement, analytics, security, and protection.

**Right Panel (Mobile App):**

\* App Service Name: AwesomeMobileApp (.azurewebsites.net)

\* Subscription: Giard

\* Resource Group: Default-Web-EastUS

\* App Service plan/Location: Default0(East US)

Pin to dashboard

**Create**

dGiard

# Creating a Mobile App

The screenshot shows the Azure portal interface for managing a mobile application named "dgtvmobile".

**Essentials:**

- Resource group: dgtv
- Status: Running
- Location: North Central US
- Subscription name: Giard
- Subscription id: b5a9a948-b91f-418c-9a14-c0d81fb5e34c
- URL: <http://dgtvmobile.azurewebsites.net>
- App Service plan/pricing tier: dgtvSvcPlan (Standard: 1 Small)
- FTP/Deployment username: dgtvmobile\diard
- FTP hostname: ftp://waws-prod-ch1-001.ftp.azurewebsites...
- FTPS hostname: https://waws-prod-ch1-001.ftp.azurewebsite...

**Monitoring:**

Requests and errors chart (Edit):

- Y-axis: 0, 20, 40, 60, 80, 100.
- X-axis: 11:45 PM, JAN 7, 12:15 AM, 12:30 AM.
- Legend: HTTP SERVER ERRORS (red), REQUESTS (blue).

**Settings:**

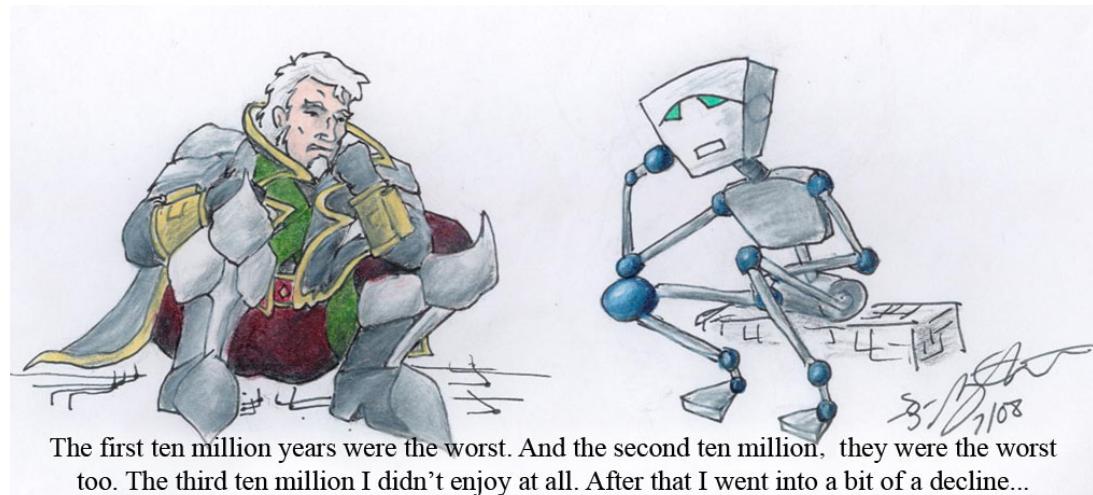
- Search settings: Search bar.
- SUPPORT & TROUBLESHOOTING:
  - Check health
  - Troubleshoot
  - New support request
- GENERAL:
  - Quick start
  - Properties
  - Application settings
- APP SERVICE PLAN:
  - App Service Plan
  - Scale Up (App Service Plan)
  - Scale Out (App Service Plan)
- MOBILE:
  - Easy tables

# REST

<https://Giard.azurewebsites.net/Tables/Table1>

<https://mymobileapp-code.azurewebsites.net/Tables/Table1>

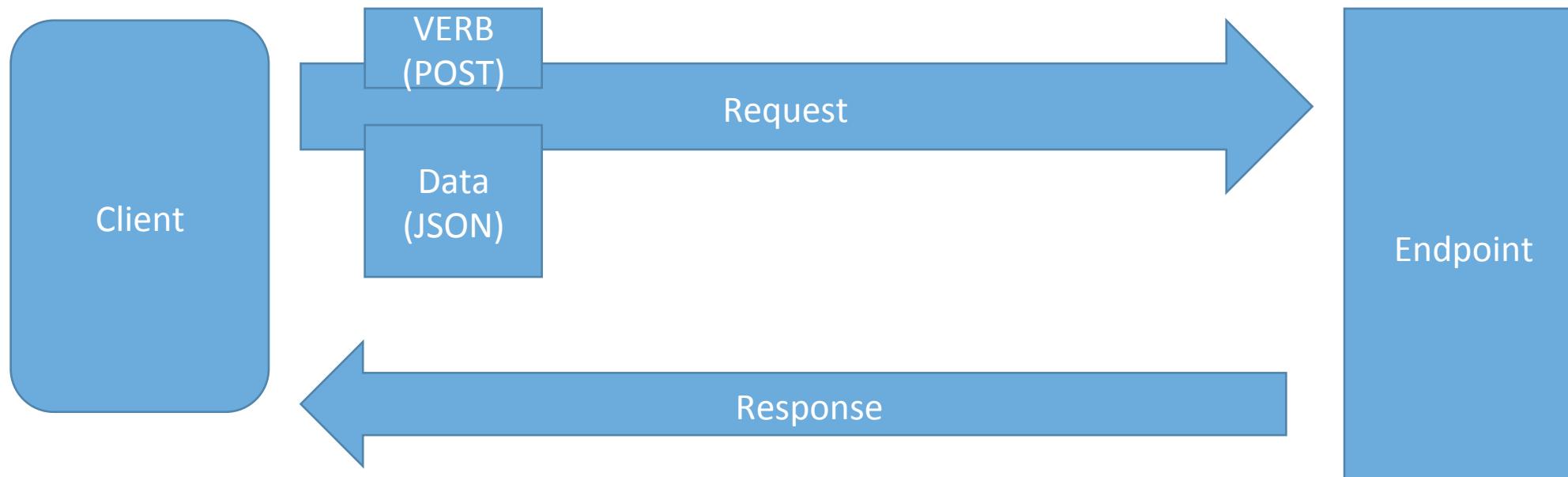
- GET, PUT, POST, DELETE, PATCH
- Extend GET with oData Query syntax



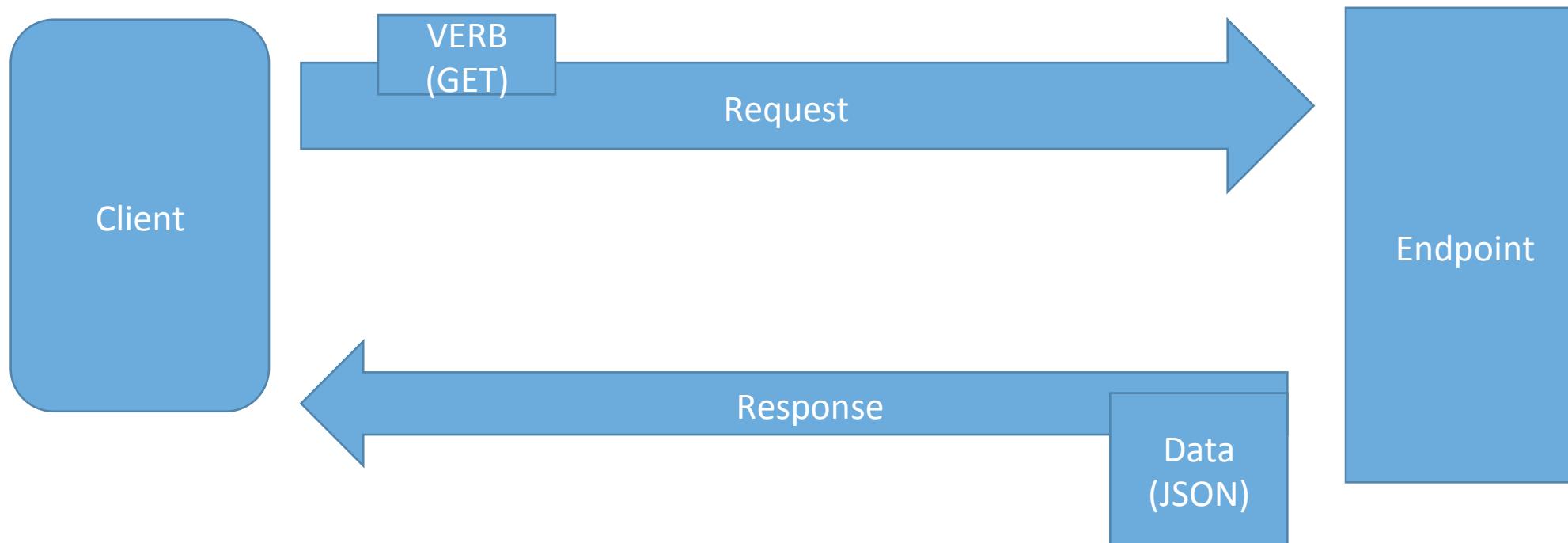
# REST

HTTP Verb	Action	SQL
GET	Read Data	SELECT...
POST	Create Data	INSERT...
DELETE	Delete Data	DELETE...
PUT or PATCH	Update Data	UPDATE...

# REST



# REST

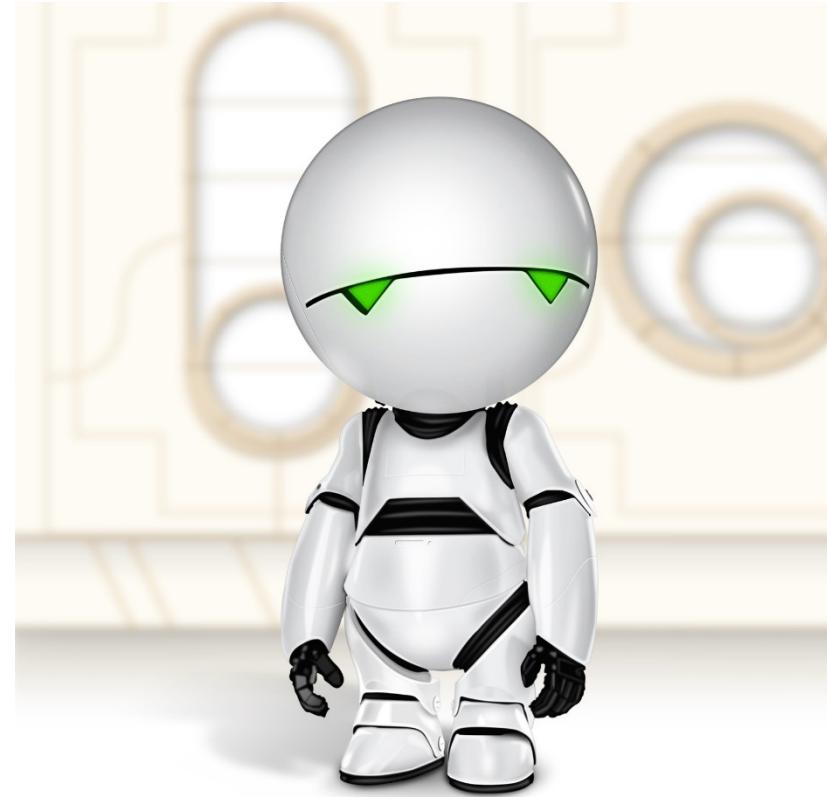


# oData Query Syntax

Extension	Description
\$filter	WHERE clause
\$inlinecount	# items in table
\$orderby	SORT clause
\$select	Columns to return
\$skip	#records to skip
\$top	#records to return

../Tables/Table1?\$filter=state eq IL

../Tables/Table1?\$filter=state eq IL&\$orderby=LastName



# Dynamic Schema



- Automatically adds columns if matching data submitted
- Useful during development
- Turn off during production

# Permissions

Change permissions

Save      Discard

Insert permission

- Allow anonymous access
- Authenticated access only
- Disabled

Allow anonymous access

Delete permission

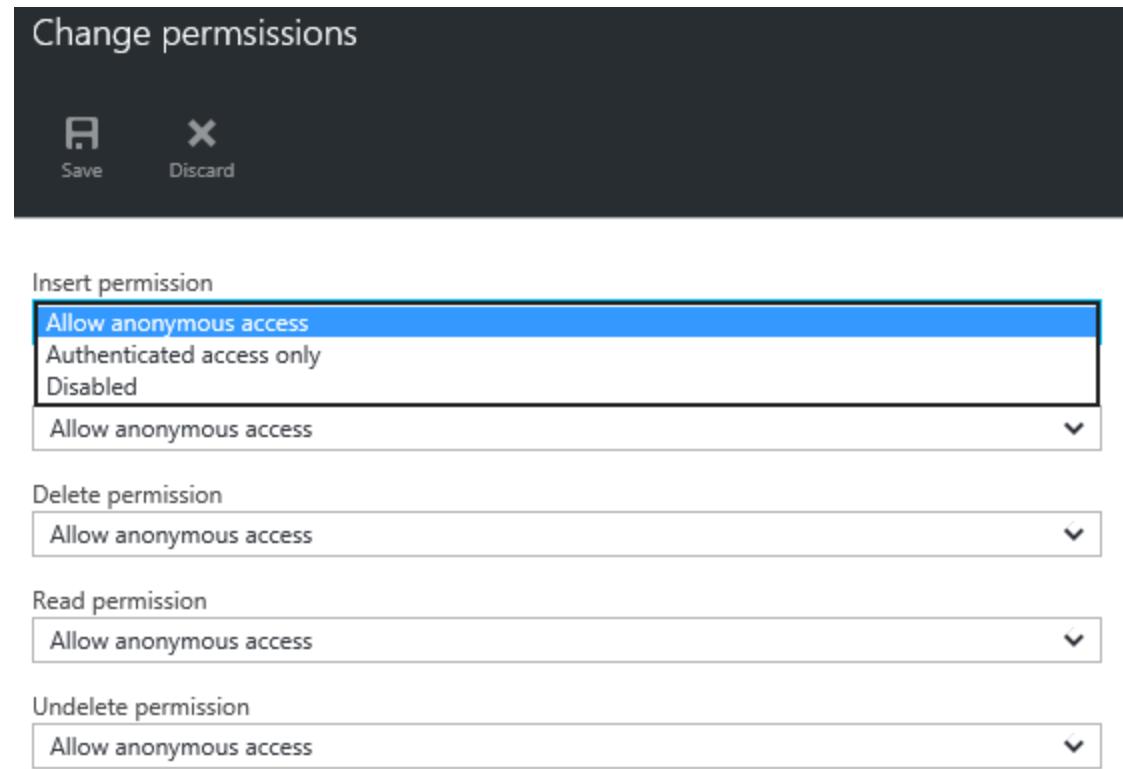
Allow anonymous access

Read permission

Allow anonymous access

Undelete permission

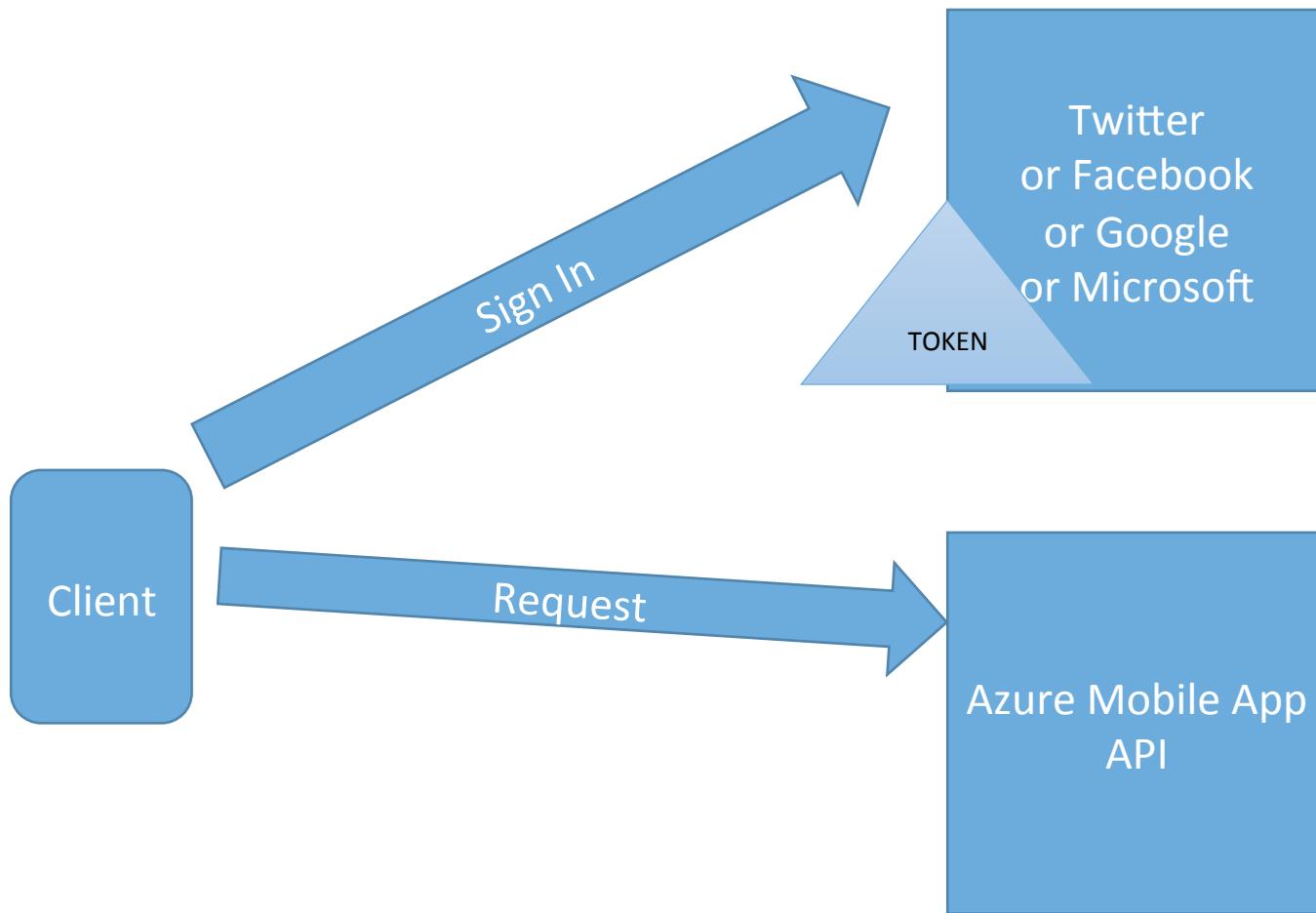
Allow anonymous access



Demo



# Identity – Single Sign-On



# Identity

 DASHBOARD DATA API SCHEDULER PUSH **IDENTITY** CONFIGURE SCALE LOGS



Your mobile service was created.  
Now let's connect it to an app.



@DavidGiard

# Identity

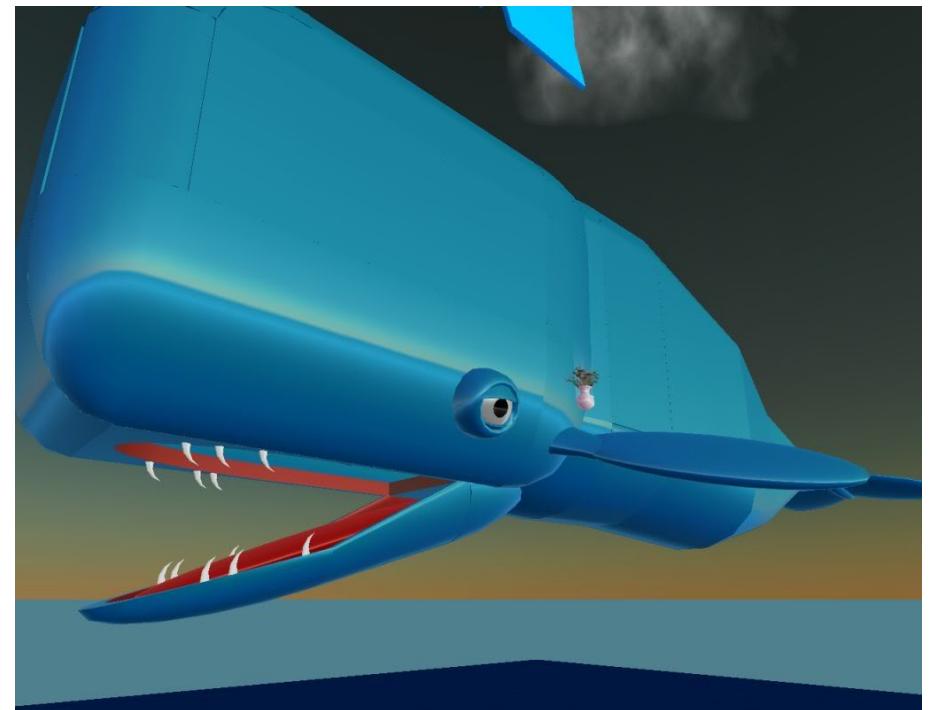
```
function insert(item, user, request) {  
    item.CreatedBy = user.userId;  
    request.execute();  
}
```

```
function read(query, user, request) {  
    query.where ({CreatedBy:user.userId});  
    request.execute();  
}
```

# Demo



# Scaling



# Scaling

Save X Discard

AVERAGE INSTANCES  
1

\* Scale by schedule and performance rules

Description Create your own set of rules. Create a schedule that adjusts your instance counts based on time and performance metrics.

Default, scale 1 - 1

Settings

- CPU Percentage > 80 (increase count by 1)
- CPU Percentage < 60 (decrease count by 1)
- Add Rule
- Add Profile

Notifications for Scale Actions

Email Administrator and CoAdministrators

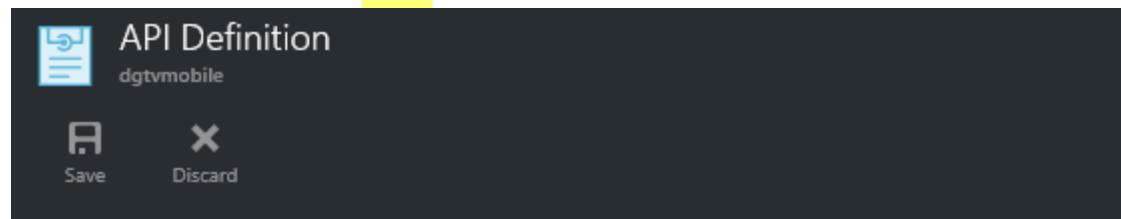
Save X Discard

- Authentication / Authorization
- Backups
- Custom domains
- SSL certificates
- Networking
- Scale up (App Service plan)**
- Scale out (App Service plan)**
- Security Scanning
- WebJobs
- MySQL In App (preview)
- Properties
- Locks
- Automation script

APP SERVICE PLAN

- App Service plan
- Quotas
- Change App Service plan

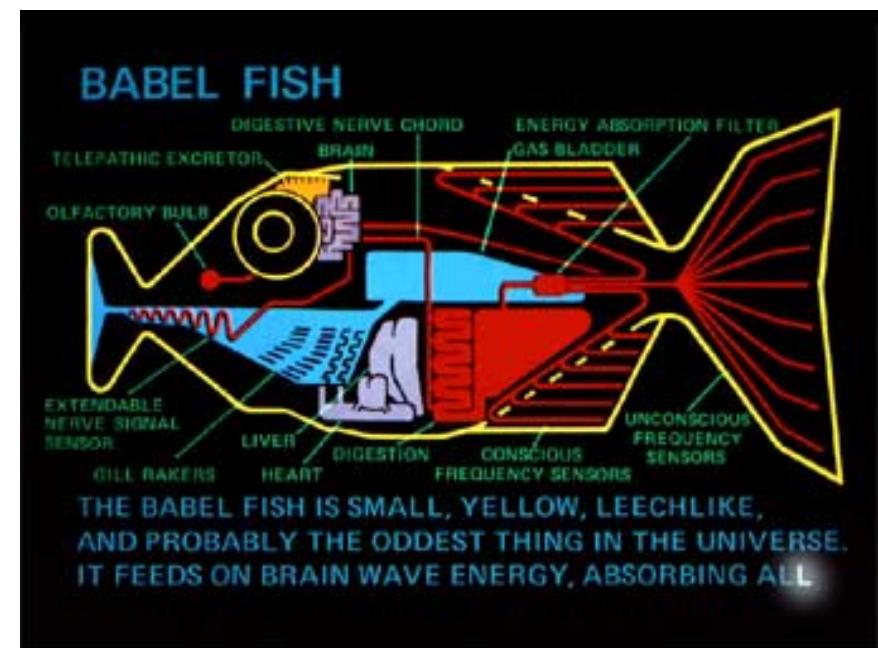
# API Scripts



**i** API definition lets you configure the location of the Swagger 2.0 metadata describing your API. This makes it easy for others to discover and consume your API. Note: the URL can be a relative or absolute path, but must be publicly accessible.

API definition location

*URL of API definition (e.g., <http://www.yoursite.com/apidefinition.json>)*



@DavidGirard

# API Scripts

<Mobile Service URL>/api/<API Name>

e.g.:

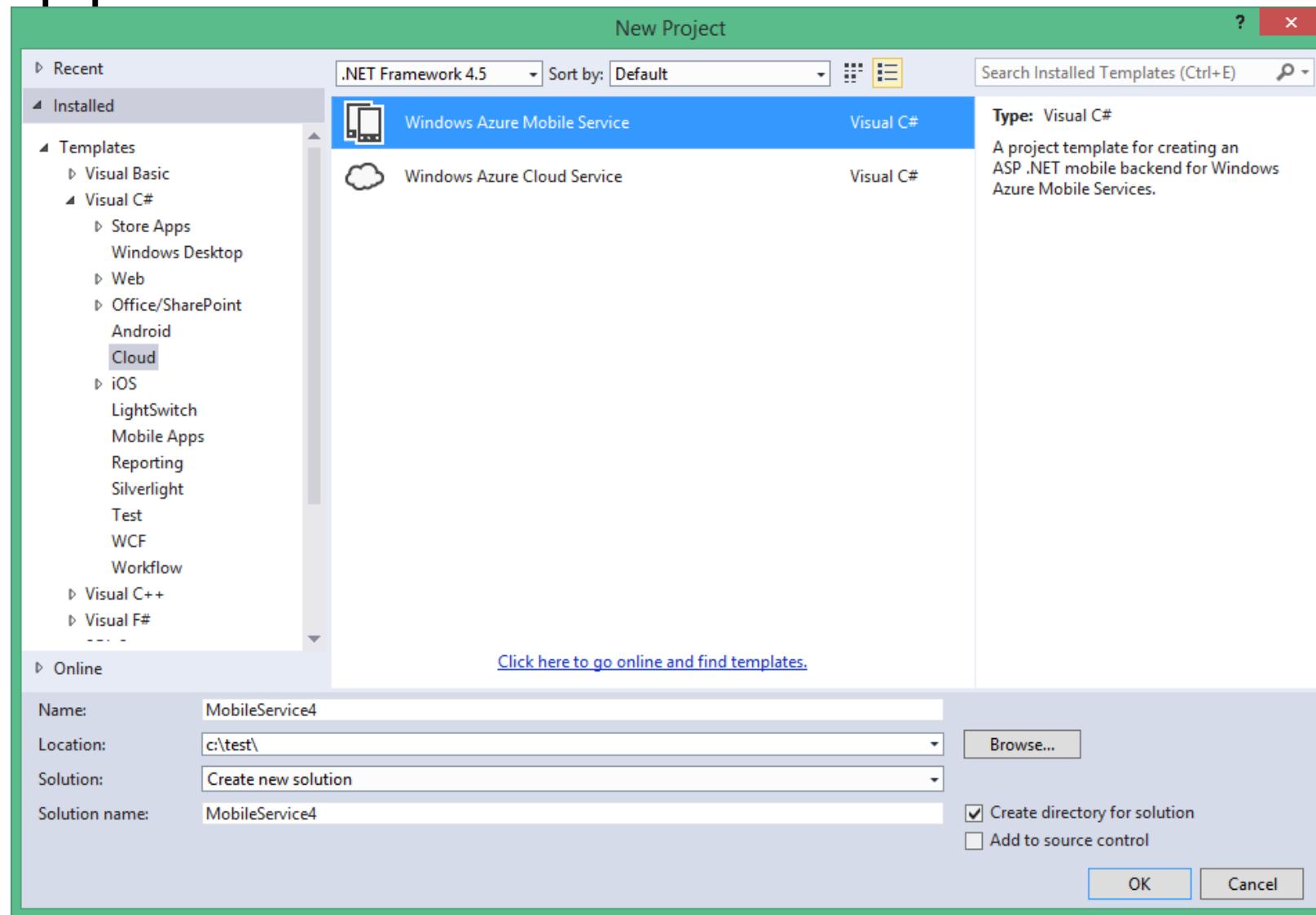
<https://giard.azure-mobile.net/api/MyApi>

```
exports.post = function(request, response) {
    // Use "request.service" to access features of your mobile service, e.g.:
    // var tables = request.service.tables;
    // var push = request.service.push;

    response.send(statusCodes.OK, { message : 'Hello World!' });
};

exports.get = function(request, response) {
    response.send(statusCodes.OK, { message : 'Hello World!' });
};
```

# Mobile Apps in .NET



# Mobile Apps in .NET

```
public class TodoItemController : TableController<TodoItem>
{
    protected override void Initialize(HttpControllerContext controllerContext)
    {
        base.Initialize(controllerContext);
        MobileService4Context context = new MobileService4Context(Services.Settings.Schema);
        DomainManager = new EntityDomainManager<TodoItem>(context, Request, Services);
    }

    // GET tables/TodoItem
    public IQueryable<TodoItem> GetAllTodoItems()
    {
        return Query();
    }

    // GET tables/TodoItem/48D68C86-6EA6-4C25-AA33-223FC9A27959
    public SingleResult<TodoItem> GetTodoItem(string id)
    {
        return Lookup(id);
    }

    // PATCH tables/TodoItem/48D68C86-6EA6-4C25-AA33-223FC9A27959
    public Task<TodoItem> PatchTodoItem(string id, Delta<TodoItem> patch)
    {
        return UpdateAsync(id, patch);
    }
}
```

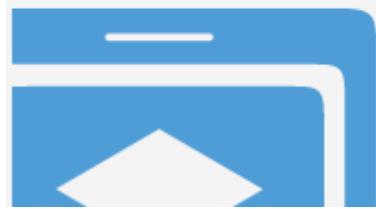
# Demo



@DavidGiard

# Push Notifications

DASHBOARD DATA API SCHEDULER PUSH IDENTITY CONFIGURE SCALE LOGS



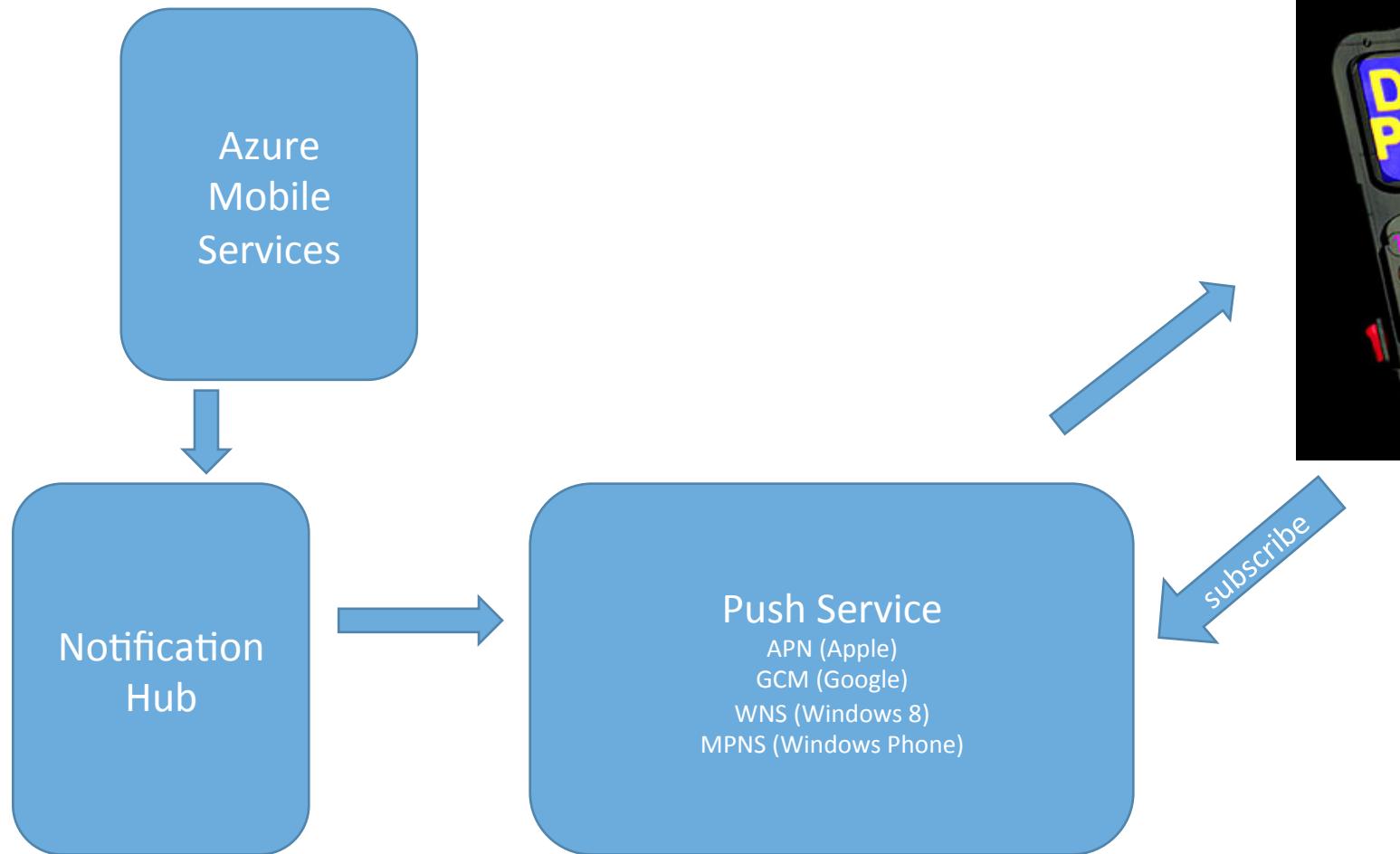
Your mobile service was created.  
Now let's connect it to an app.



# Push Notifications

```
var payload =  
  '<?xml version="1.0" encoding="utf-8"?><toast><visual><binding template="ToastText01">' +  
  '<text id="1">Sample Toast</text></binding></visual></toast>';  
var push = request.service.push;  
  
push.wns.send(null,  
  payload,  
  'wns/toast', {  
    success: function (pushResponse) {  
      console.log("Sent push:", pushResponse);  
    }  
  });  
}
```

# Push Notifications



[tinyurl.com/PushNoti](http://tinyurl.com/PushNoti)

# Push Notification on Windows 8 & WP8

1. Create Mobile Service
2. Write Client App
  1. Associate app with store
  2. Get Package SID and Client ID from Live Services. Copy to Mobile Service.
  3. Register notifications channel in OnLaunched (App.xaml.cs)
  4. Enable Toast notifications (Package.appxmanifest)
3. Update service to send Push Notification

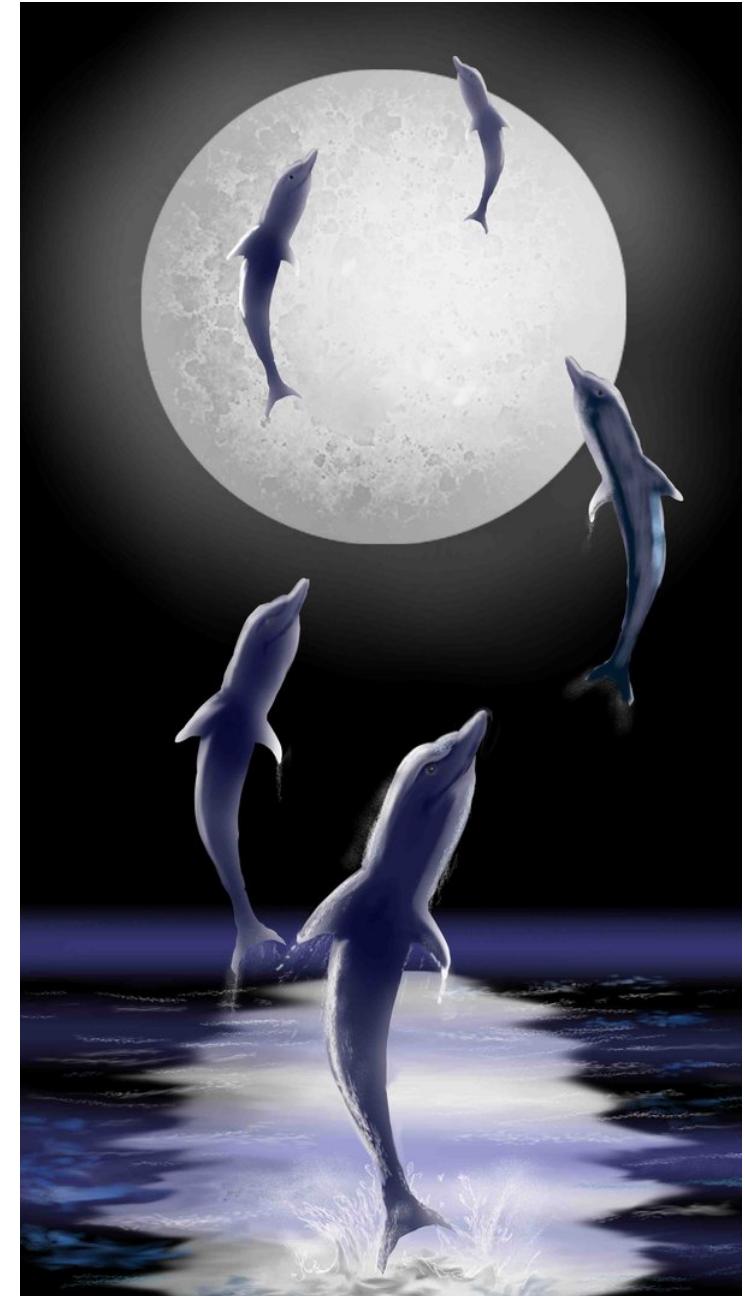
[tinyurl.com/Win8Push](http://tinyurl.com/Win8Push)

@DavidGiard

# So Long and Thanks!

David Giard

- @DavidGiard
- DavidGiard.com
- TechnologyAndFriends.com
- channel9.msdn.com/niners/dgiard



@DavidGiard