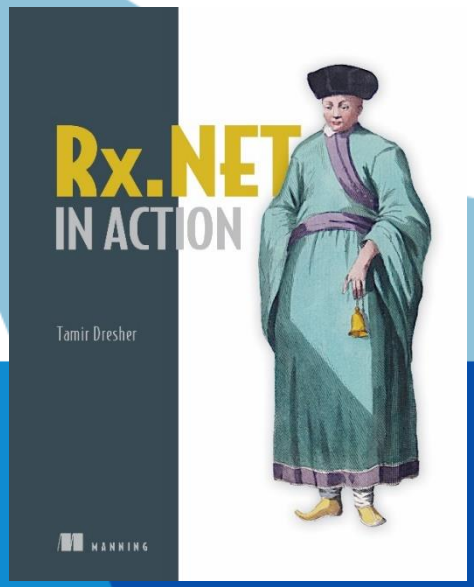
Azure Cloud Camp February 2017

App Services

Tamir Dresher (@tamir_dresher)

Senior Software Architect

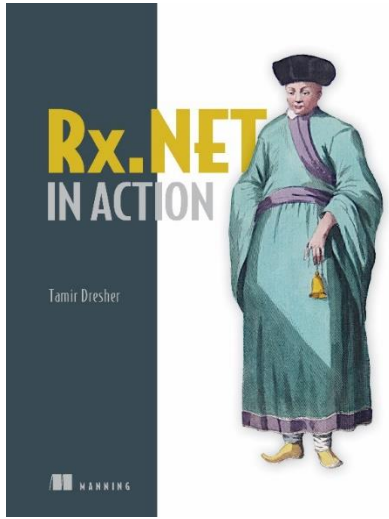


About Me

@tamir_dresher
tamirdr@codevalue.net
http://www.TamirDresher.com.

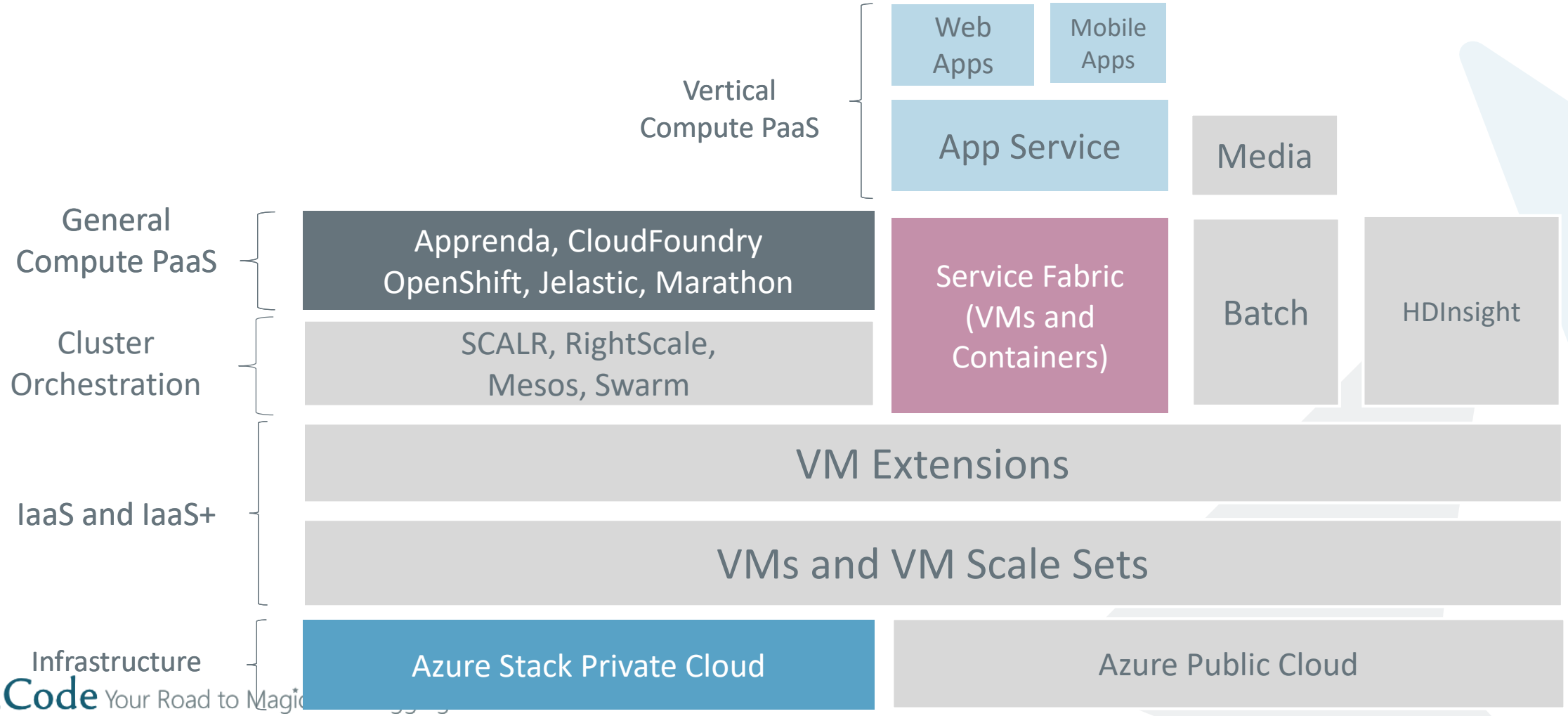
OzCode Your Road to Magical Debugging

- Author of *Rx.NET in Action* (manning publications)
- Software architect, consultant and instructor
- Software Engineering Lecturer @ Ruppin Academic Center
- Expert in large-scale, server-side, highly-concurrent systems
- Member of Microsoft Azure Advisors group



App Services

- Next level of PaaS
- Easy deployment, including Continuous Delivery



https://github.com/CodeValue/AzureCloudCampFeb2017

Agenda

- Web Apps
  - WebJobs
- App Service Plans
- Deployment Slots
- Mobile Apps
- API Apps
- Logic Apps
- Azure Functions

Azure Web Apps

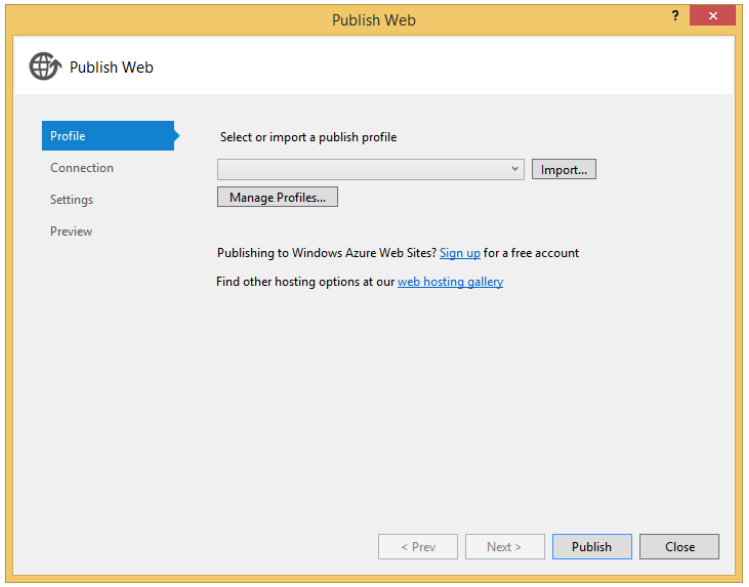
Azure Web Apps

- Provision a Web Application Fast
- You can use IDE, PowerShell, Portal
- Deploy Easily via a Source Control

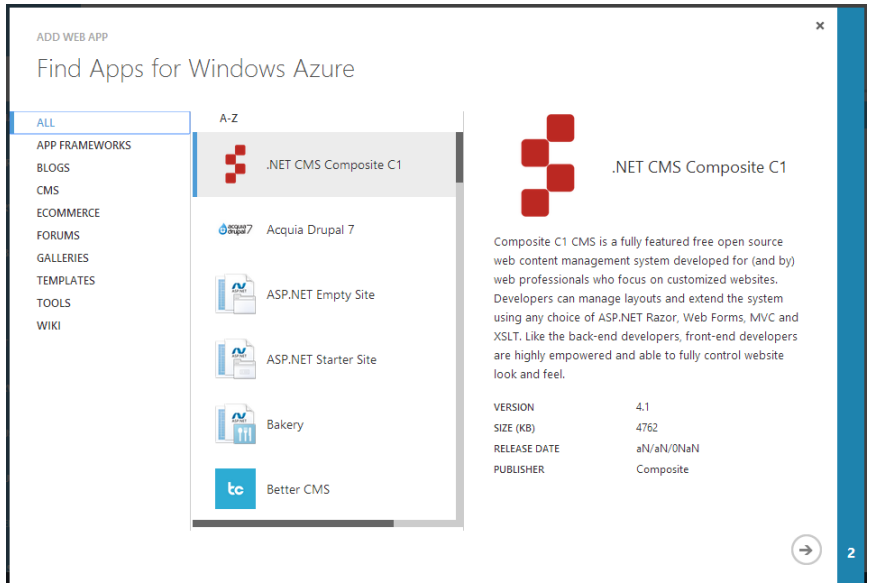
https://github.com/CodeValue/AzureCloudCampFeb2017

Creating a Web App

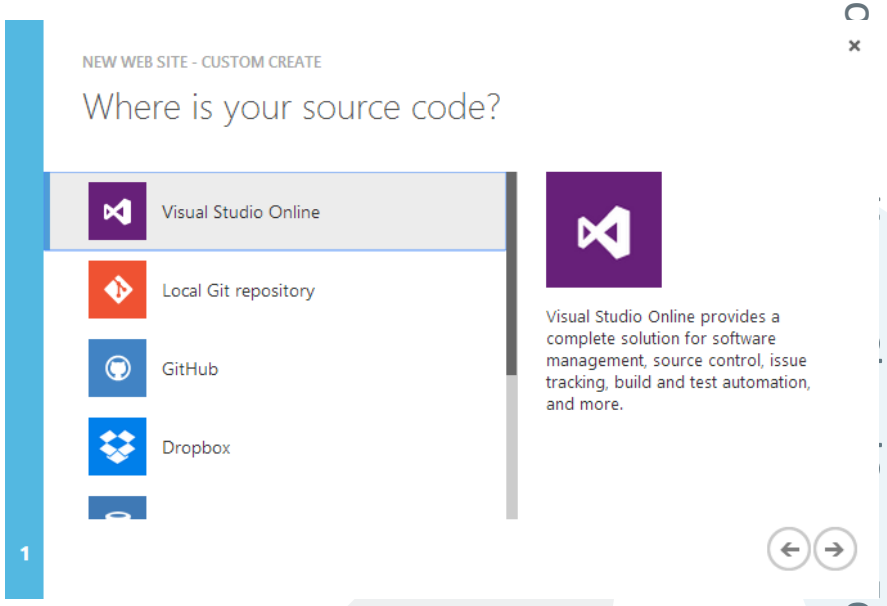
Publish from VS



From Gallery



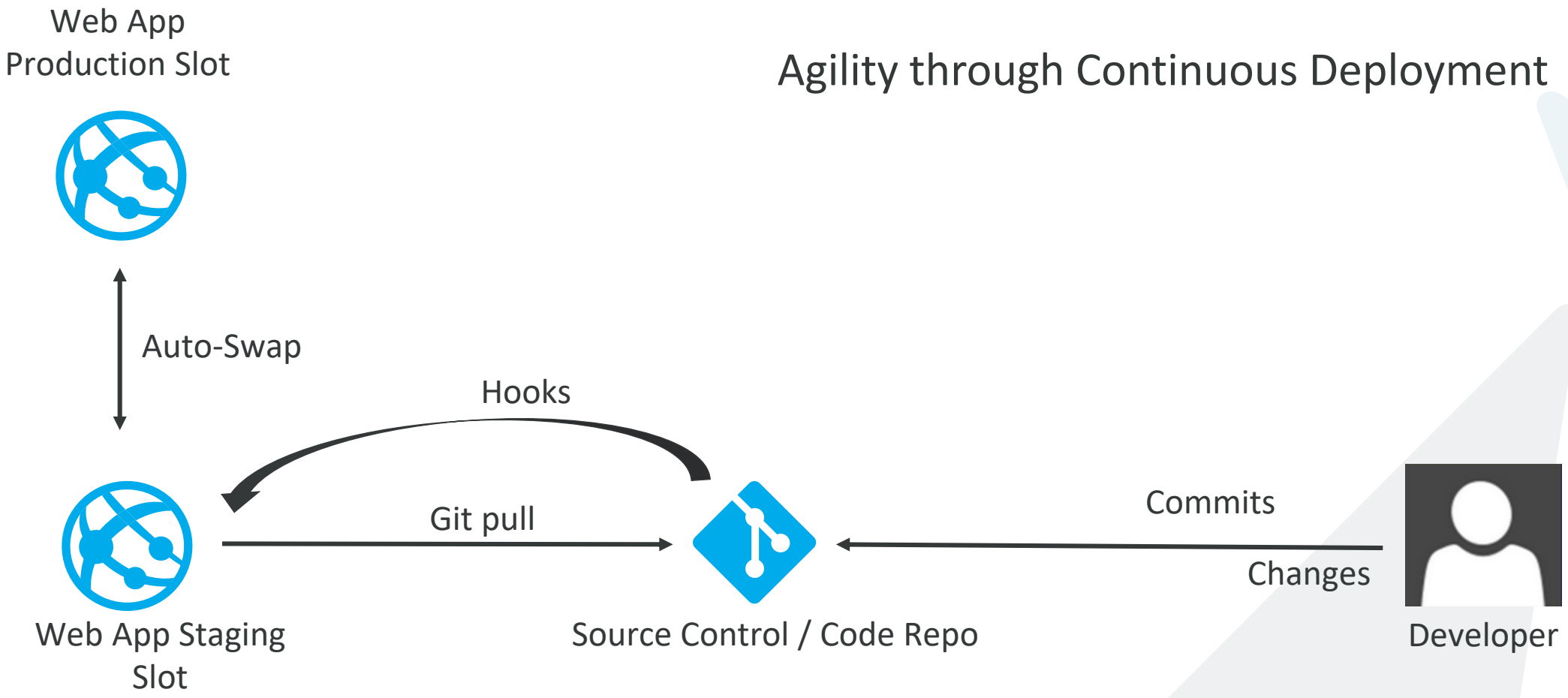
Sync with Source Control



https://github.com/Cc

eb2017

Continuous Deployment for Web Apps



Agility through Continuous Deployment

https://github.com/CodeValue/AzureCloudCampFeb2017

Continuous Deployment for Web Apps

All resources > internalwebapp1 > Settings > Deployment source > Choose source

Settings

Change App Service plan

PUBLISHING

Deployment slots

Deployment source

Deployment credentials

API

API definition

CORS

MOBILE

Easy tables

Easy APIs

Push

Deployment source

Set up deployment source

* Choose Source
Configure required settings

Performance Test
Not Configured

OK

Choose source

Visual Studio Team Services
By Microsoft

OneDrive
By Microsoft

Local Git Repository
By Git

GitHub
By GitHub

Bitbucket
By Atlassian

Dropbox
By Dropbox

External Repository

https://github.com/CodeValue/AzureCloudCampFeb2017

DEMO

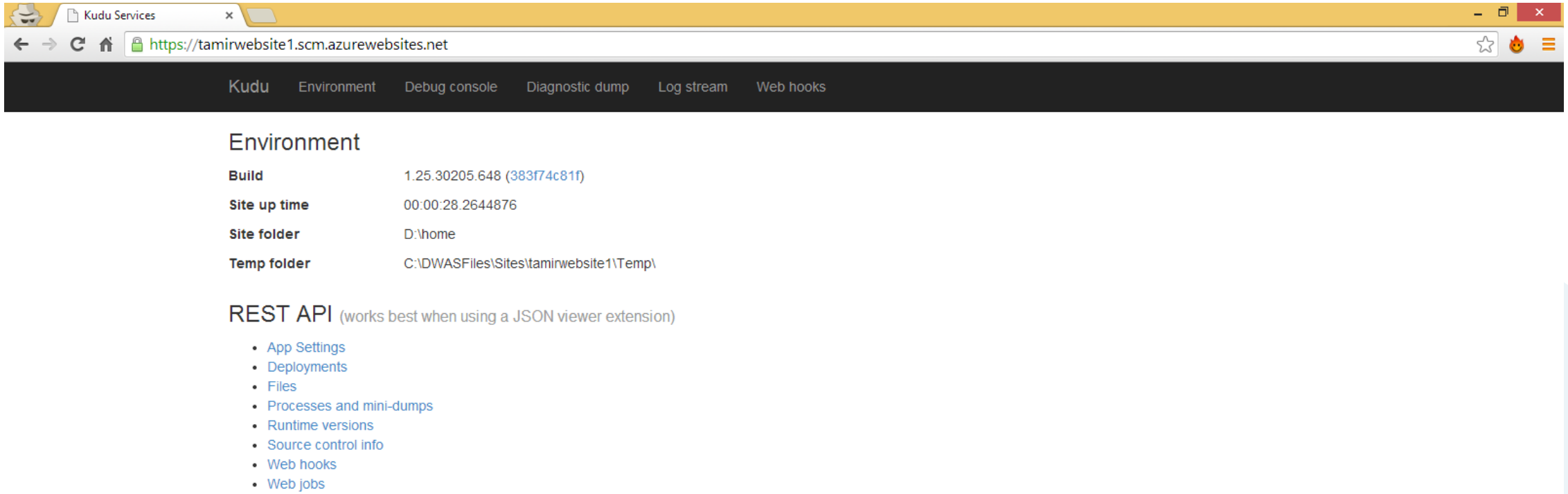
Creating a Web App from
Source Control

kudu

- Every Azure Web App has an associated Kudu service site.
- Kudu is the engine behind git deployments in Azure Web Sites
- If your web site has URL
  http://mysite.azurewebsites.net/
  then the root URL of the Kudu service is
  https://mysite.**scm**.azurewebsites.net/.
- Gives monitoring utils for the deployment

https://github.com/CodeValue/AzureCloudCampFeb2017

kudu



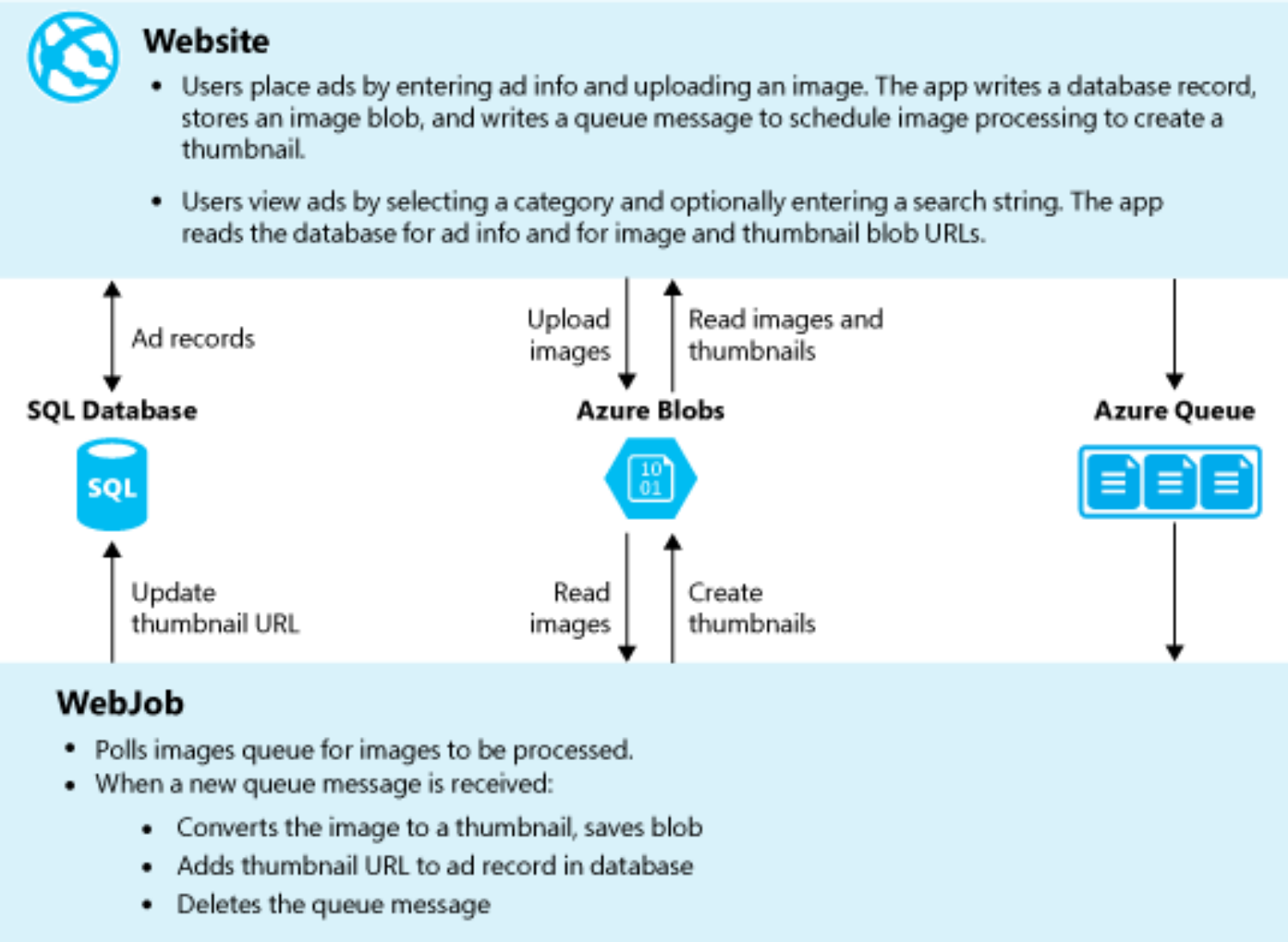
https://github.com/CodeValue/AzureCloudCampFeb2017

WebJobs

- Windows Azure Web App enables you to run custom jobs (running executables or scripts) on your web site
- The WebJobs SDK has a binding and trigger system which works with Windows Azure Storage Blobs, Queues and Tables.
- The trigger system calls a function in your code whenever any new data is received in a queue or blob.
- You can create your own binders and triggers

WebJobs – Typical Scenarios

- Image processing or other CPU-intensive work.
- Queue processing.
- RSS aggregation. If you have a site that maintains a list of RSS feeds, you could pull in all of the articles from the feeds in a background process.
- File maintenance, such as aggregating or cleaning up log files.
- Other long-running tasks that you want to run in a background thread, such as sending emails..



WebJobs – Execution

- Continuously - For programs that need to be running all the time, such as services that poll queues.
  - Runs all instances if Always On* configuration setting is enabled
- Schedule - For programs that need to be run at particular times, such as nightly file maintenance tasks
  - Runs on a single instance selected for load balancing by Microsoft Azure.
- Manual/On demand/WebHook - When you want to start a program manually, such as when you want to start an additional run of a file maintenance task outside its normal schedule
  - Runs on a single instance selected for load balancing by Microsoft Azure.

Coding with WebJobs SDK

- Install the nuget package: Microsoft.Azure.WebJobs
  - There are extensions you can also use: Microsoft.Azure.WebJobs.Extensions, Microsoft.Azure.WebJobs.Extensions.WebHooks
- Write methods for the background tasks that you want to execute, and you decorate them with attributes from the WebJobs SDK.

```
public static void ProcessQueueMessage([QueueInput("webjobsqueue")] string inputText,
                                       [BlobOutput("containername/blobname")]TextWriter writer)
{
    writer.WriteLine(inputText);
}
```

- In the Main method create and start JobHost

```
static void Main()
{
    JobHost host = new JobHost();
    host.RunAndBlock();
}
```

https://github.com/CodeValue/mpF

Coding a WebJob

- The framework looks for any public static methods that have WebJobs SDK attributes
- The Framework watches for the triggers for those methods, such as new queue messages or new blobs. When a triggering event occurs, the framework calls the method.

https://github.com/CodeValue/AzureCloudCampFeb2017

WebJobs Triggers and Binders

- **QueueInput** attribute means that this method will be called when a queue message is received
- **BlobInput** attribute means the method will be called when a new blob appears in a specified container
  - By default, the WebJobs SDK looks for connection strings named AzureWebJobsStorage and AzureWebJobsDashboard but you can change

https://github.com/CodeValue/AzureCloudCampFeb2017

WebJobs - example

- Let's say I want to take this function that works fine at the command line and run it in the cloud at scale.

```
public static void SquishNewlyUploadedPNGs(Stream input, Stream output)
{
    var quantizer = new WuQuantizer();
    using (var bitmap = new Bitmap(input))
    {
        using (var quantized = quantizer.QuantizeImage(bitmap))
        {
            quantized.Save(output, ImageFormat.Png);
        }
    }
}
```

https://github.com/CodeValue/AzureCloudCampFeb2017

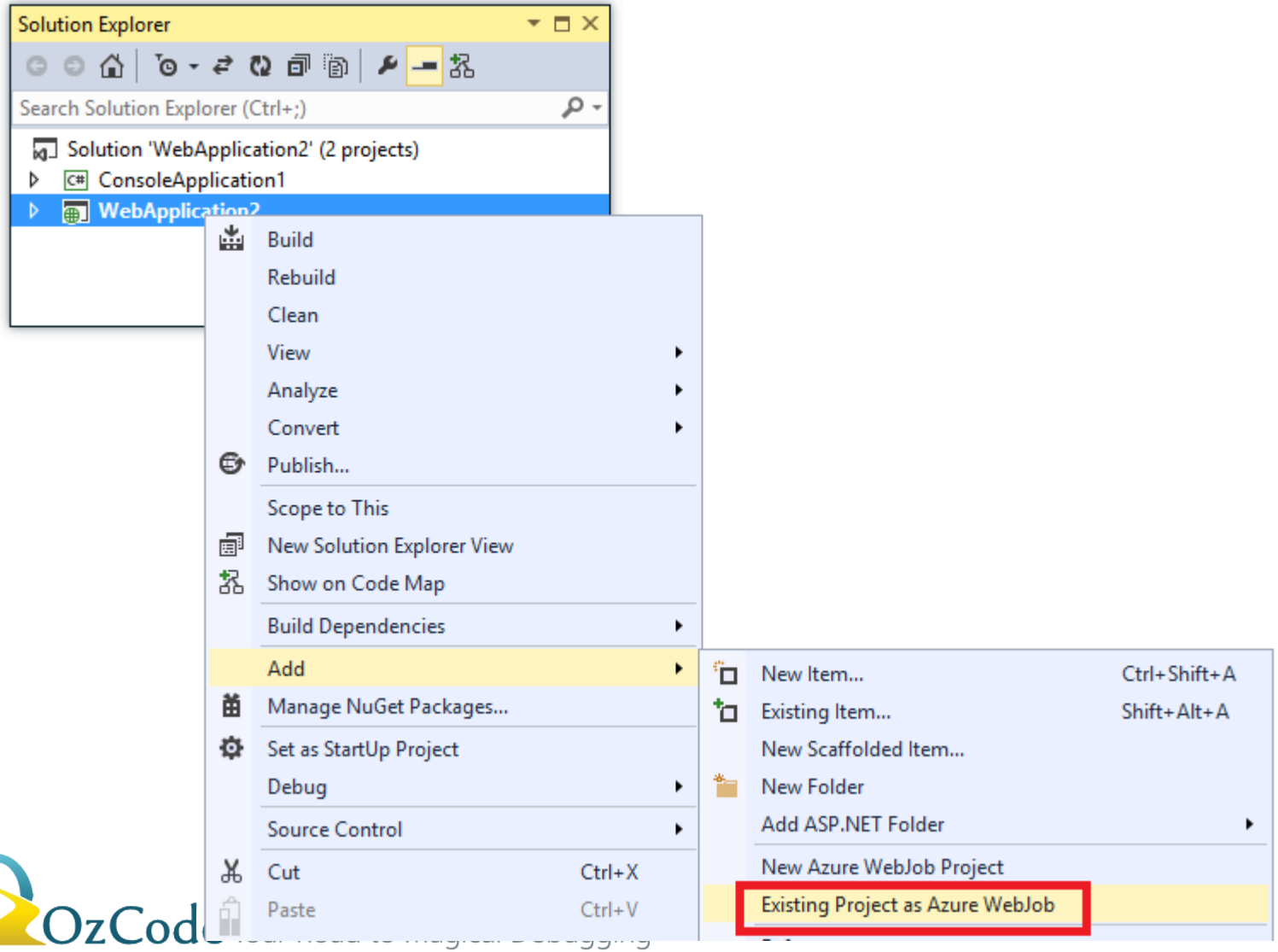
WebJobs - example

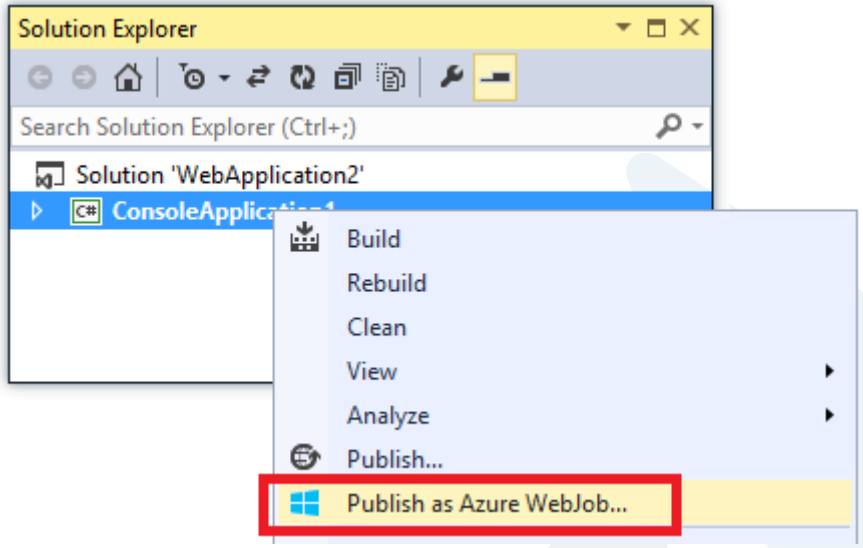
- Let's say I want to take this function that works fine at the command line and run it in the cloud at scale.

```
class Program
{
    static void Main(string[] args)
    {
        JobHost host = new JobHost();
        host.RunAndBlock();
    }

    public static void SquishNewlyUploadedPNGs(
        [BlobInput("input/{name}")] Stream input,
        [BlobOutput("output/{name}")] Stream output)
    {
        var quantizer = new WuQuantizer();
        using (var bitmap = new Bitmap(input))
        {
            using (var quantized = quantizer.QuantizeImage(bitmap))
            {
                quantized.Save(output, ImageFormat.Png);
            }
        }

    }
}
```

https://github.com/CodeValue/AzureCloudCampFeb2017

Publishing a WebJob





https://github.com/CodeValue/AzureCloudCampFeb2017

webjob-publish-settings.json

- Visual Studio installs the Microsoft.Web.WebJobs.Publish NuGet package for WebJob projects
- the scheduling information is stored in a *webjob-publish-settings.json* file in the project *Properties* folder
- http://schemastore.org/schemas/json/webjob-publish-settings.json

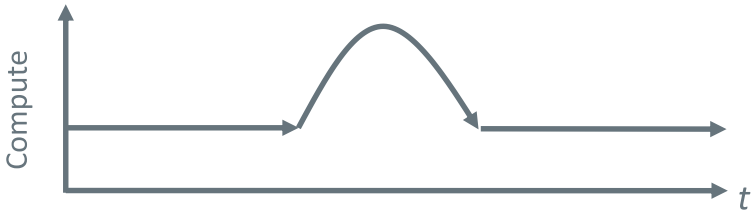
App Service Scaling

Azure App Services

Scaling - Cloud Computing Patterns

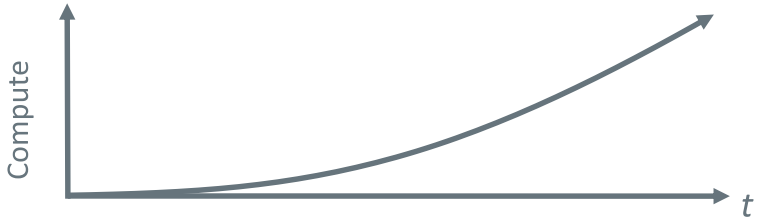
On and Off

On & off workloads (e.g. batch job)
Over provisioned capacity is wasted
Time to market can be cumbersome



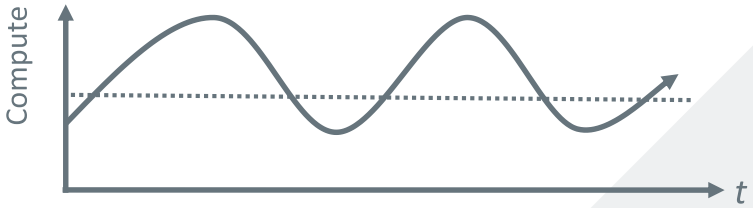
Unpredictable Bursts

Unexpected/unplanned peak in demand
Sudden spike impacts performance
Can't over provision for extreme cases



Growing Fast

Successful services needs to grow/scale
Keeping up w/ growth is big IT challenge
Cannot provision hardware fast enough



Predictable Bursts

Services with micro seasonality trends
Peaks due to periodic increased demand
IT complexity and wasted capacity

Scaling Up vs. Scaling Out

Scale Up

Vary the VM size

1 Core w/ 1.75 GB RAM
2 Cores w/ 3.5 GB RAM
4 Cores w/ 7 GB RAM

Scale Out

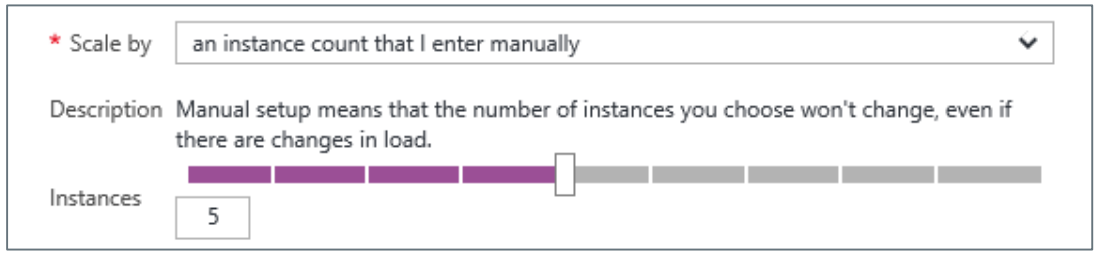
Vary the VM count

Max 3 instances*
Max 10 instances
*Max 20/50** instances*

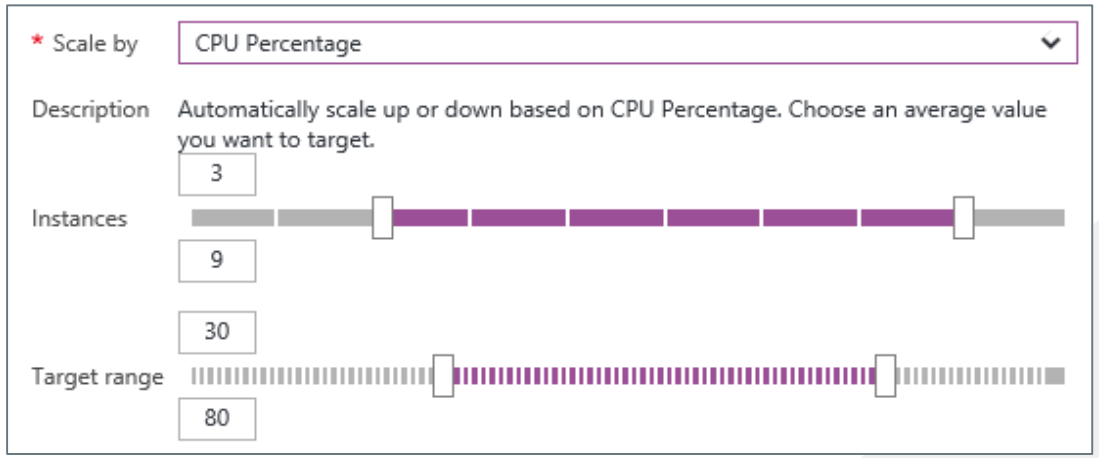
https://github.com/CodeValue/AzureCloudCampFeb2017

Manual Scaling vs. Auto-Scaling

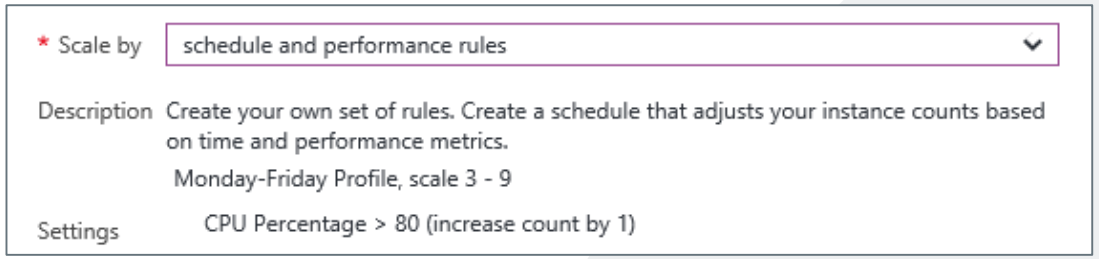
Manual – Scale via portal or scripts



Auto – CPU Percentage



Auto – Schedule & Performance Rules



https://github.com/CodeValue/AzureCloudCampFeb2017

App Service Plan

- Represents a set of physical resources that can be shared across multiple apps in Azure App Service.
- 5 pricing tiers – Free, Shared, Basic, Standard, Premium
- Apps can share the Service Plan if they are in the same subscription and same location
- A good usage for example is to share resources for each environment (DEV, TEST, PROD)

App Service App

App Service App

App Service App

App Service Plan

SKU: Premium (# of sites, storage, slots…)

Compute Resource: P4 (8 cores, 14 GB memory)

Scale: 8

App Service App

App Service App

App Service App

App Service Plan

SKU: # of sites, storage, slots…

Compute Resource: cores, memory

Scale: number of instances

https://github.com/CodeValue/AzureCloudCampFeb2017

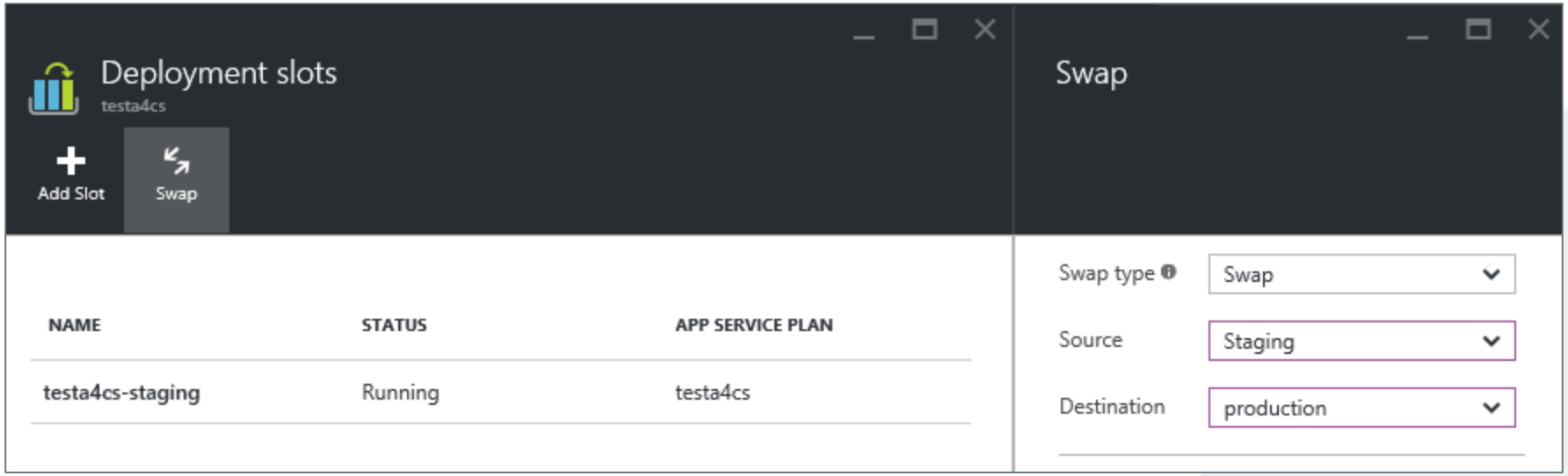
App Service Plans

- Billing and provisioning for App Service resources

	Free	Shared	Basic	Standard	Premium
# of Apps	10	100	Unlimited	Unlimited	Unlimited
Shared Disk Space	1 GB	1 GB	10 GB	50 GB	500 GB
Maximum Instances	1	1	3	10	50
Autoscale	No	No	No	Yes	Yes
Staging Environments				5	20
Custom Domains	No	Yes	Yes	Yes	Yes
SLA			99.95%		

Deployment Slots

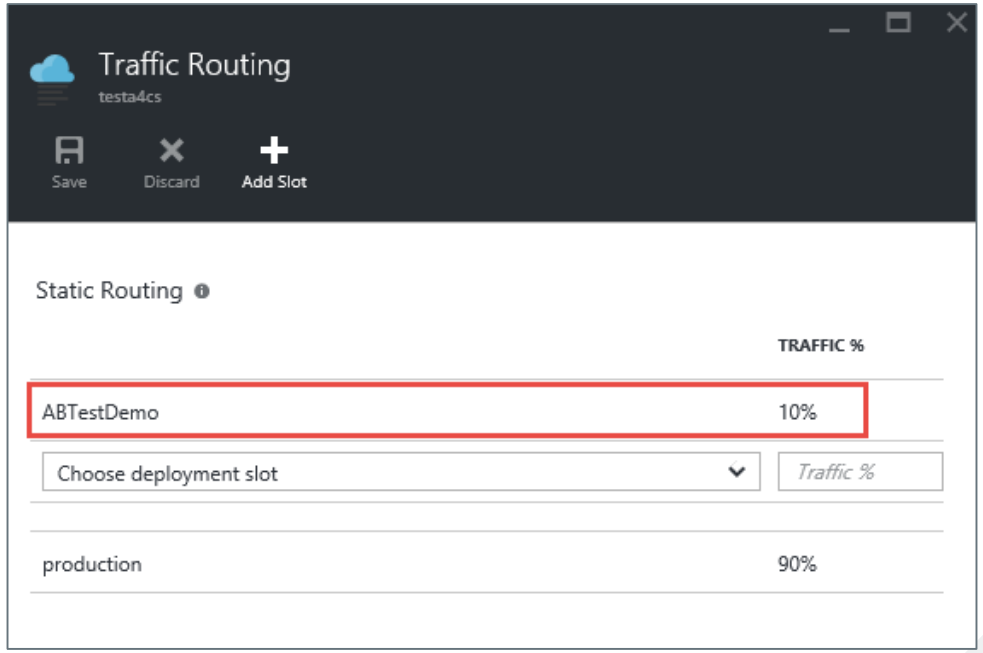
- Use a Deploy-Confirm-Promote workflow
  - Promote via “swap” through Azure portal
- http://sitename-slotname.azurewebsites.net



https://github.com/CodeValue/AzureCloudCampFeb2017

A/B Testing

- Test changes by routing requests to different deployment slots
- Use Traffic Routing to direct % of traffic to alternate slots

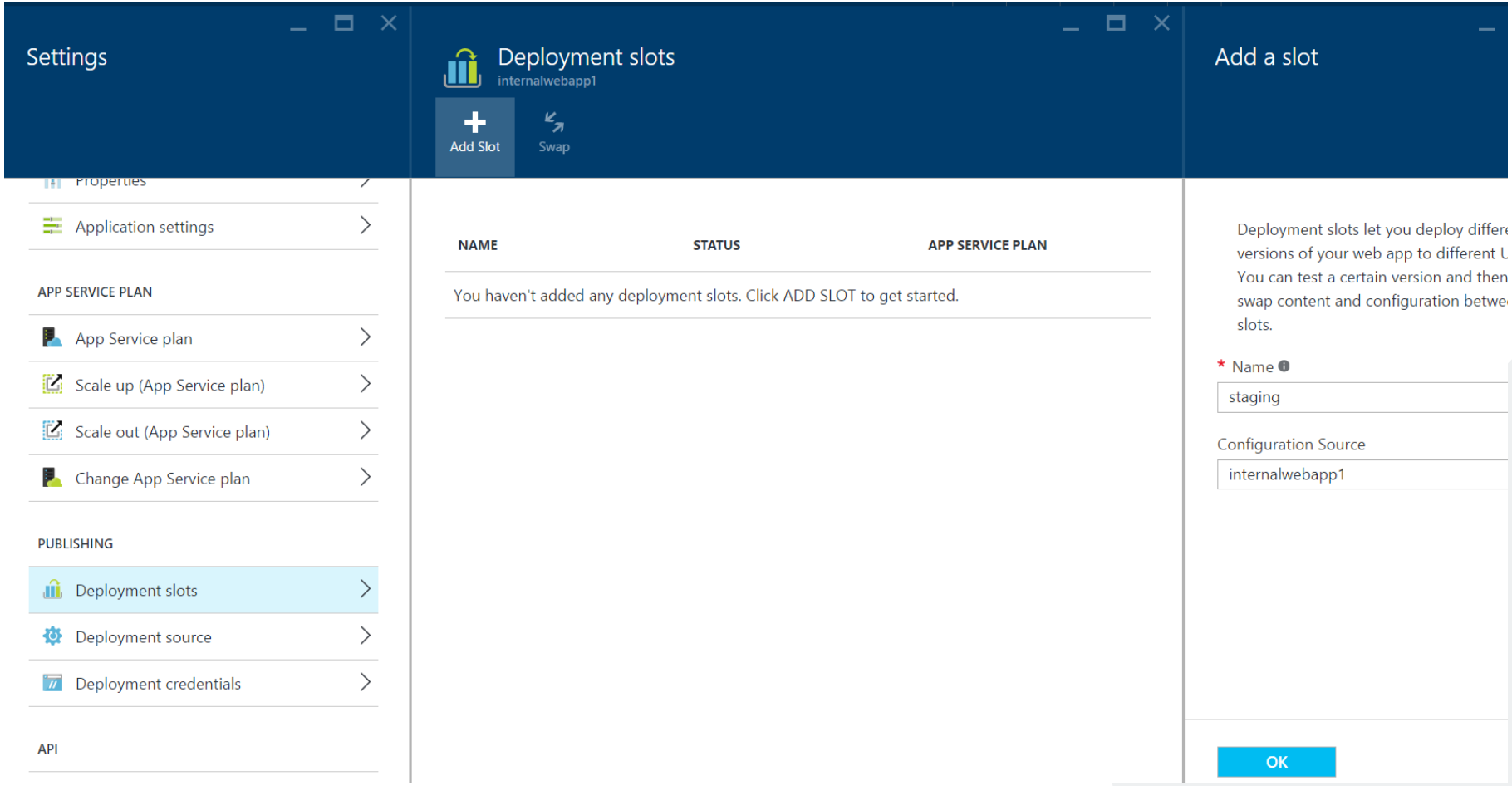


https://github.com/CodeValue/AzureCloudCampFeb2017

Deployment slots

Deployment Slots

- Provide different deployment environments
  - Only for Standard and Premium App Service Plans



https://github.com/CodeValue/AzureCloudCampFeb2017

Deployment Slots App Settings

Settings

Site Metrics per instance
App Service plan Metrics per instance
Mitigate
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)
Change App Service plan

Application settings
internalwebapp1-staging

Save Discard

Remote debugging Off On

Remote Visual Studio version 2012 2013 2015

App settings

WEBSITE_NODE_DEFAULT_V...	4.2.3	Slot setting
SLOT_NAME	STAGING	Slot setting
Key	Value	Slot setting

Connection strings

No results

Name	Value	SQL Database	Slot setting

Default documents

Default.htm

Default.html

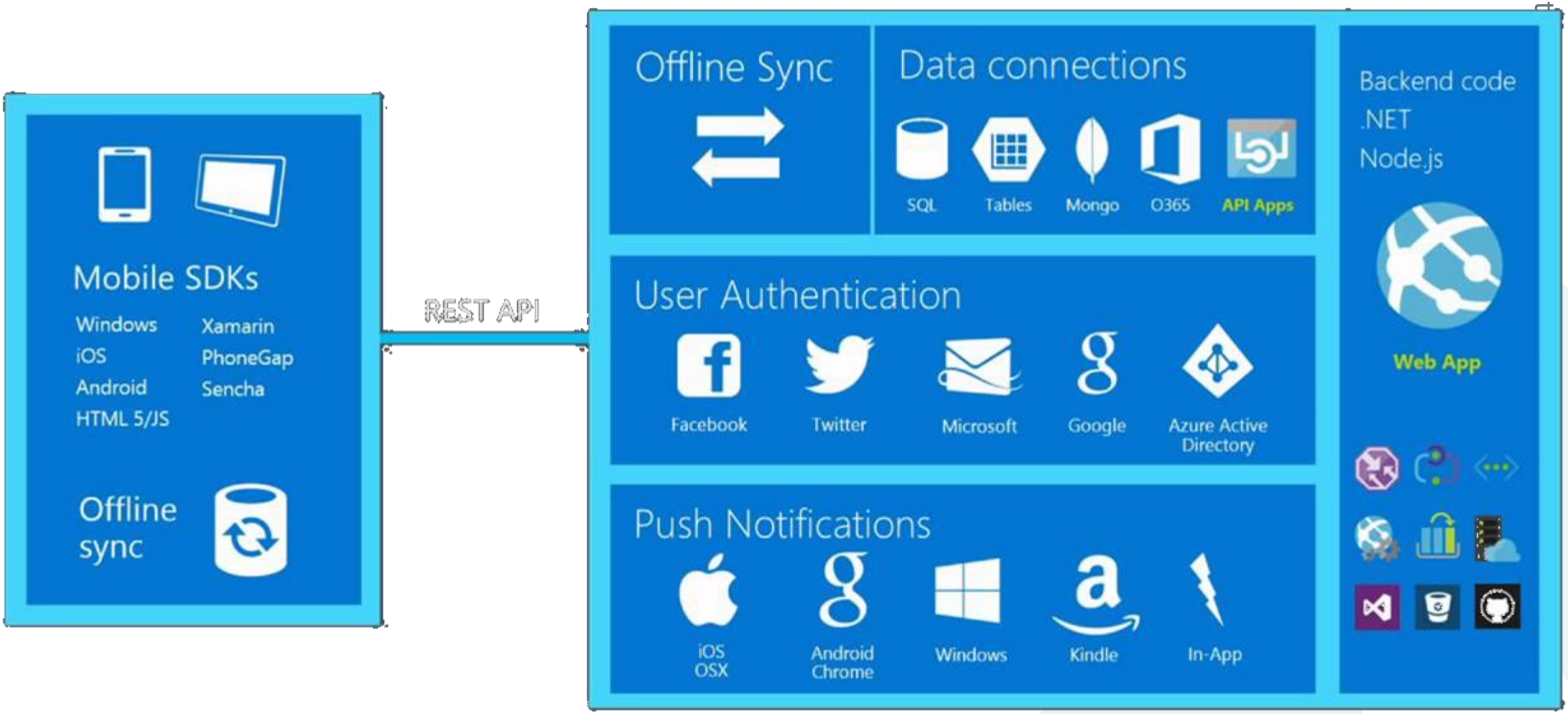
https://github.com/CodeValue/AzureCloudCampFeb2017

Mobile Apps

- Azure App Services

https://github.com/CodeValue/AzureCloudCampFeb2017

What is Mobile Apps?



https://gi

Structured Storage

- Powered by SQL Database
- Same DB – Multiple Mobile Services
- Data management in
  - Windows Azure Portal
  - SQL Portal
  - SQL Management Studio
  - REST API
  - CLI Tools
- JSON to SQL Type Mappings

The REST API

Base REST API Endpoint URL

https://Mobileservice.azure-mobile.net/tables/*

Data Operations and their REST Equivalents

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

Server Side Scripts

- Customizing logic on the server

Node.js scripts
.NET (Preview)

Intercept CRUD requests to tables

Passes through to SQL by default

Fully customizable logic flow

https://github.com/CodeValue/AzureCloudCampFeb2017

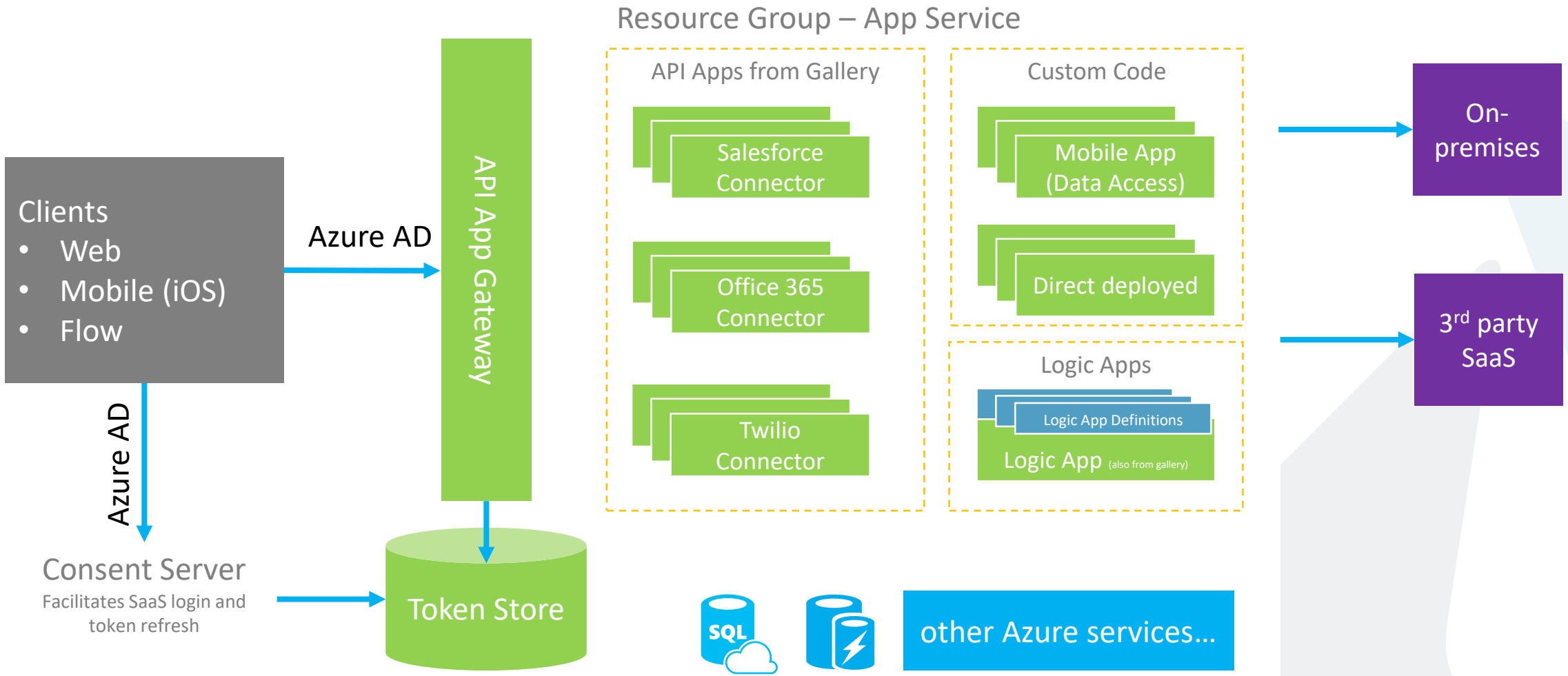
API Apps

Azure App Services

API Apps

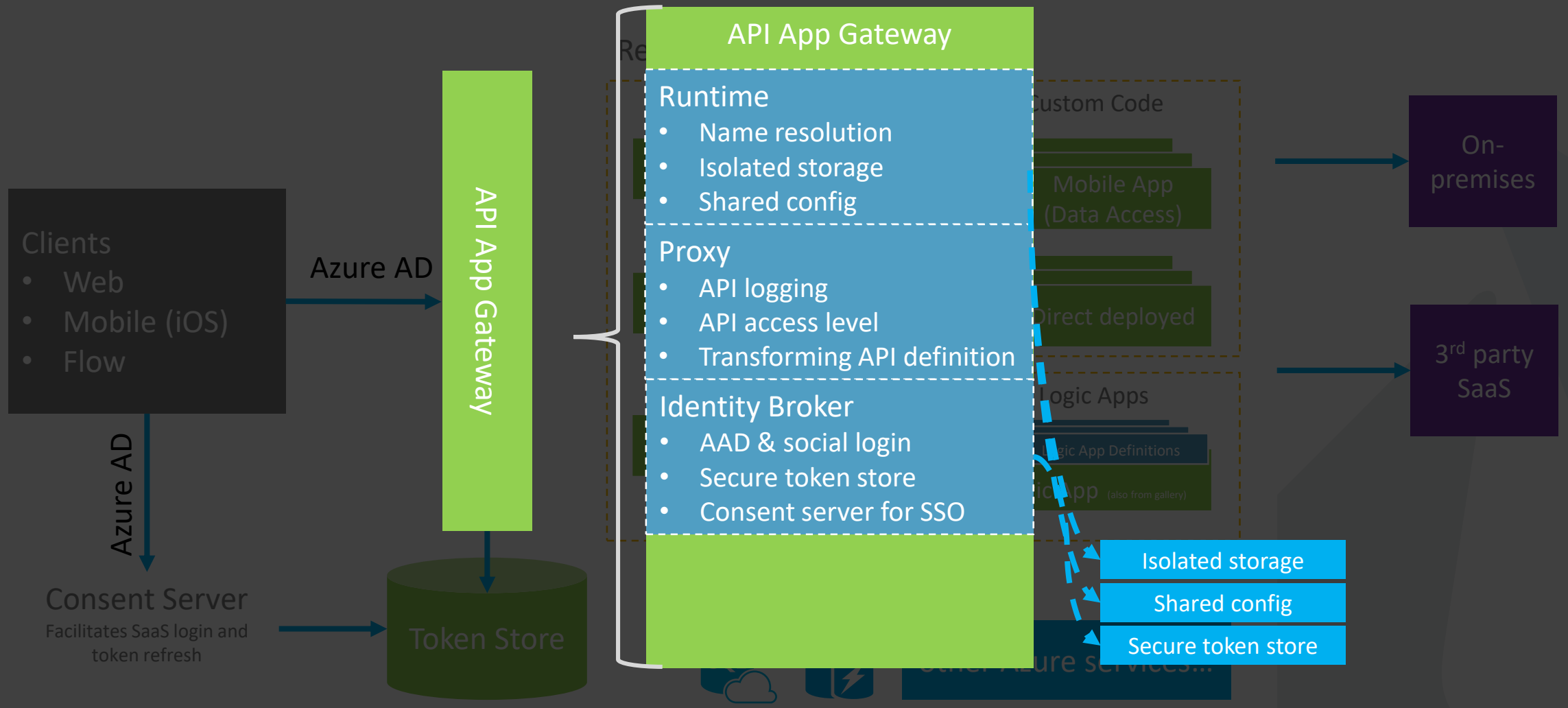
- Basically Web Apps for Web Api's
- Simple access control
- Swagger metadata
- Logic App Integration
- Marketplace support for connectors
- VS tooling and support (for client side as well)

API Apps Architecture Example



https://github.com/CodeValue/AzureCloudCampFeb2017

API Apps Architecture Example



Logic Apps

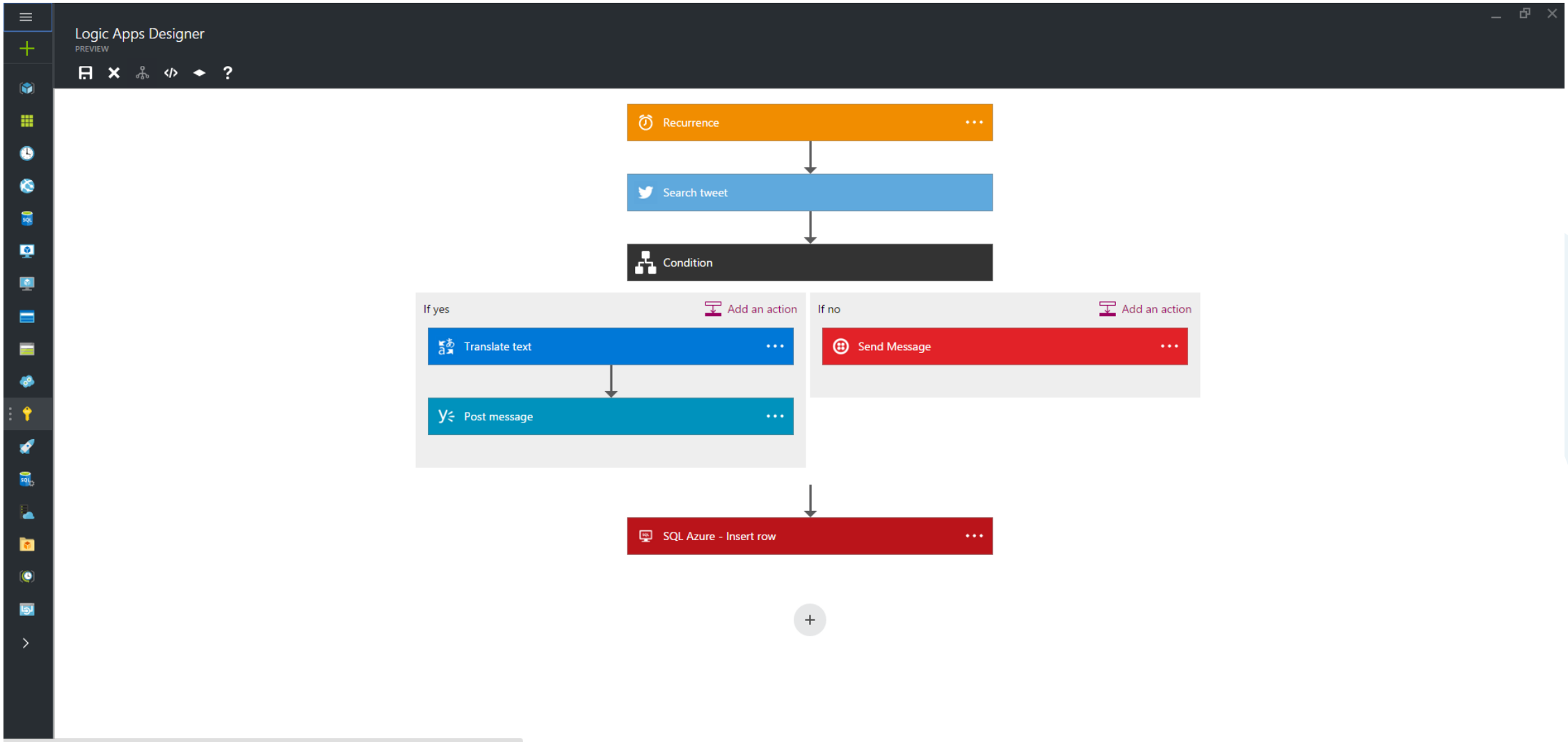
Azure App Services

Logic Apps

- Visually create business process and workflows based on Triggers and Actions
- Deliver integration capabilities in Web, Mobile, and API Apps
- Integrate with your SaaS and enterprise applications
- Automate EAI/B2B and business processes
- Connect to on-premises data

DEMO – Creating Logic App

Logic App Designer



https://github.com/CodeValue/AzureCloudCampFeb2017

Azure Functions

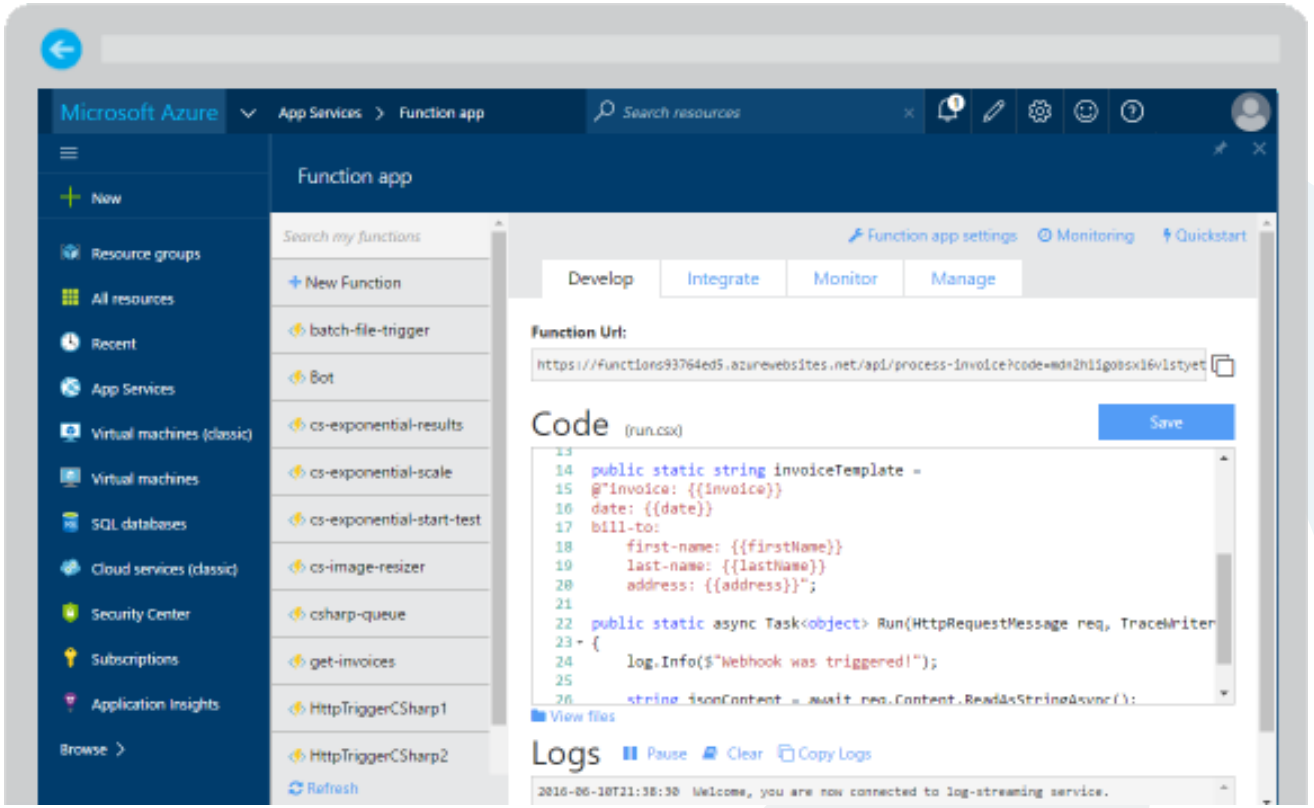
Azure Functions

- Azure Functions is an event driven, compute-on-demand experience
- Azure Functions scale based on demand and you pay only for the resources you consume.
- Function can be written in C# or nodeJS
- The runtime, otherwise known as the script host, is the underlying WebJobs SDK host which listens for events, gathers and sends data, and ultimately runs your code.

https://github.com/CodeValue/AzureCloudCampFeb2017

Azure Functions

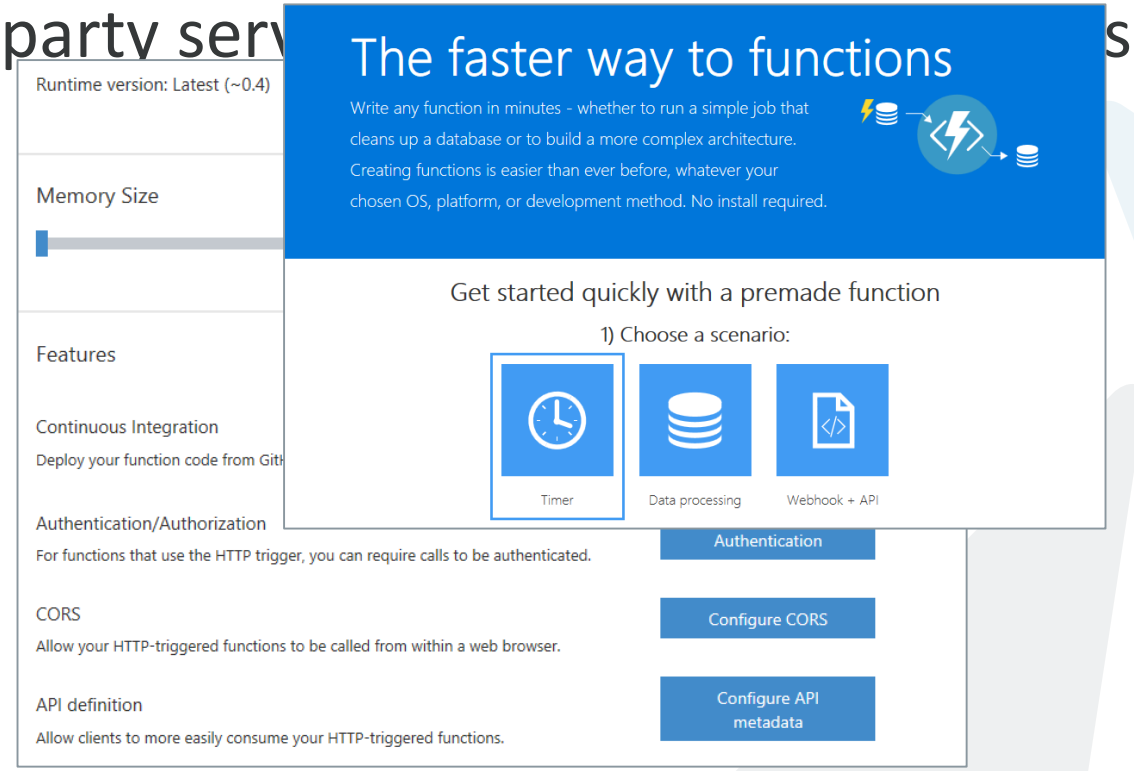
Create a “serverless” event-driven experience that extends the existing Azure App Service platform by building “nanoservices” that can scale based on demand



https://github.com/CodeValue/AzureCloudCampFeb2017

Supported Languages and Tools

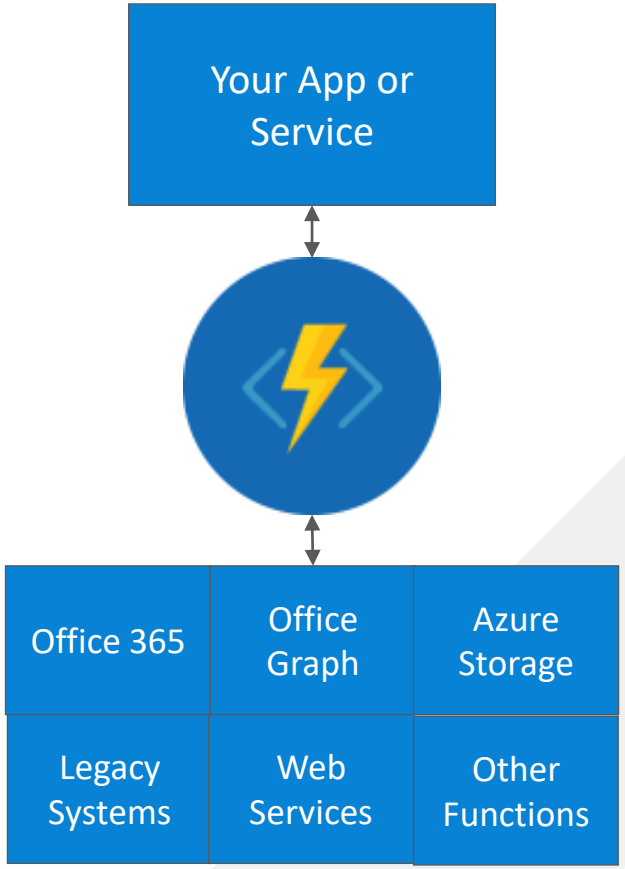
Create functions in JavaScript, C#, Python, and PHP, as well as scripting options such as Bash, Batch, and PowerShell, that can be triggered by virtually any event in Azure, 3rd party serv s



https://github.com/CodeValue/AzureCloudCampFeb2017

Common Scenarios

- Timer-based processing
- Azure service event processing
- SaaS event processing
- Serverless web application architectures
- Serverless mobile backends
- Real-time stream processing
- Real-time bot messaging



https://github.com/CodeValue/AzureCloudCampFeb2017

Function App Templates

Function App templates are categorized into general areas of Timer, Data Processing, and Webhook & API

- BlobTrigger
- EventHubTrigger
- Generic webhook
- GitHub webhook
- HTTPTrigger
- QueueTrigger
- ServiceBusQueueTrigger
- ServiceBusTopicTrigger
- TimerTrigger
- Blank & Experimental

Choose a template

Language: All Scenario: Core

BlobTrigger - C#	BlobTrigger - Node	Empty - C#	Empty - Node
A C# function that will be run whenever a blob is added to a specified container	A Node.js function that will be run whenever a blob is added to a specified container	An empty C# function without triggers, inputs, or outputs	An empty Node.js function without triggers, inputs, or outputs
EventHubTrigger - Node	**Generic Webhook - C#**	**Generic Webhook - Node**	**GitHub WebHook - C#**
A Node.js function that will be run whenever an event hub receives a new event	A C# function that will be run whenever it receives a webhook request	A Node.js function that will be run whenever it receives a webhook request	A C# function that will be run whenever it receives a GitHub webhook request

https://github.com/CodeValue/AzureCloudCampFeb2017

Timer Function Apps

- Run at explicitly specified intervals, like every day at 2:00 am using CRON expressions, like “0 */5 * * * *“ (every 5 minutes)
- Can send information to other systems, but typically don’t “return” information, only write to logs
- Great for redundant cleanup and data management
- Great for checking state of services
- Can be combined with other functions

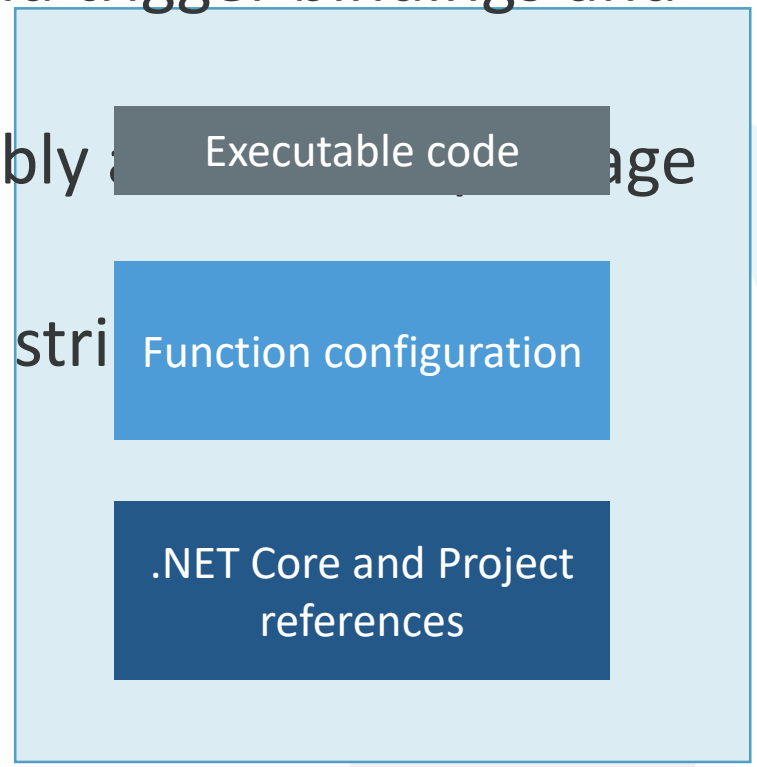
Webhook & API Function Apps

- Triggered by events in other services, like GitHub, Team Foundation Services, Office 365, OneDrive, Microsoft PowerApps
- Takes in a request and sends back a response
- Often mimic Web API and legacy web services flows
- Typically need CORS settings managed
- Best for exposing functionality to other apps and services
- Great for building Logic Apps

https://github.com/CodeValue/AzureCloudCampFeb2017

Anatomy of a Function

- A “Run” file that containing the function code
- A “Function” file containing all service and trigger bindings and parameters
- A “Project” file containing project assembly a age references
- App Service settings, such as connection stri



https://github.com/CodeValue/AzureCloudCampFeb2017

Function Bindings

Bindings serve as the basis for all connections to and from a function. Many bindings can be “bi-directional” as well.

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	✓		✓
Tables	Azure Storage		✓	✓
Tables	Azure Mobile Apps		✓	✓
No-SQL DB	Azure DocumentDB		✓	✓
Push Notifications	Azure Notification Hubs			✓

https://github.com/CodeValue/AzureCloudCampFeb2017

logging

- To log output to your streaming logs in C#, you can include a TraceWriter typed argument. We recommend that you name it **log** or **logger**. It's recommend to avoid using Console.Write in Azure Functions.

```
 public static void Run(string myBlob, TraceWriter log)
{
    log.Verbose($"C# Blob trigger function processed: {myBlob}");
}
```

https://github.com/CodeValue/AzureCloudCampFeb2017

CSX

- A lightweight C# script
- Only the .NET Framework 4.6 is supported
- If you need to reference a private assembly, you can upload the assembly file into a bin folder relative to your function and reference it by using the file name
  - `#r "AssemblyName“`
- Supports Nuget by adding the `packages.json`
  - When you upload a *project.json* file, the runtime gets the packages and automatically adds references to the package assemblies
- Other *.csx files can be reused by adding #load "myfile.csx"