Issue 25 | July-Aug 2016

DNCMagazine

www.dotnetcurry.com

EDITORIAL

Anniversaries inevitably lead to introspection and retrospection. When we started the DNC Magazine in 2012, we had a simple goal in mind - to help you to learn, prepare and stay ahead of the curve. Thus far, I believe we have done a good job and I hope our detailed walkthroughs, good programming practices and peeks into MS and JavaScript technologies have directly or indirectly helped you to improve your programming skills.

As Hellen Keller once quoted - "Alone we can do so little, together we can do so much". I have proudly repeated this time and again, that I am part of a team comprising of MVPs and experts who are terrifically tireless, extraordinarily excellent, and spectacular beyond words. From the bottom of my heart, I want to thank all my fellow authors and reviewers for their time, energy and dedication. A shout-out to our sponsors as well who have helped us keep this magazine free of cost.

The most important Thank You of all goes to you, the reader. May you continue to inspire us for many years to come! And may you always remember how much your views and feedback are needed, considered and valued! Reach out to us on twitter with our handle @dotnetcurry or email me at suprotimagarwal@dotnetcurry.com. Happy Learning!

Editor in Chief
Suprotim Agarwal
@suprotimagarwal

CREDITS

Editor In Chief
Suprotim Agarwal
suprotimagarwal@a2zknowledgevisuals.com

Art Director
Minal Agarwal
minalagarwal@a2zknowledgevisuals.com

Contributing Authors
Damir Arh
Kunal Chandratre
Ravi Kiran
Shoban Kumar
Subodh Sohoni
Suprotim Agarwal
Vikram Pendse
Yacoub Massad

Technical Reviewers
Damir Arh
Gil Fink
Suprotim Agarwal
Yacoub Massad

Next Edition
1st September 2016

CONTENTS

Subodh Sohoni

Being Agile in the World of Fixed Bid Projects

Most software companies worldwide are adopting Agile Development techniques. Many of these software companies do contractual software development. Agility and contractual development against a fixed bid are conceptually two ends of a spectrum. In this article, we will see an option for companies to be agile even when they are executing a contractual fixed bid project.

Agile is here to stay. Everybody wants to be Agile including businesses that depend on the contractual development of software (also called as outsourcing). The style of development for contractual development and agile teams are a lot different. Imagine a golfer with a handicap below five , aspiring to play golf in the style of Samba soccer. Simply put, being Agile and also bidding for a fixed bid project, just don't gel.

OK, let's first understand what Agile and Fixed bid projects are.

Agile Development

The Agile team takes things as it is thrown at them. They don't plan on how to develop a software until they develop it. This scenario is like crossing a bridge only when you reach it. You don't plan how you will cross the bridge until you get to the bridge. Many environmental conditions, constraints, and decisions can change before you reach the bridge. It may happen that by the time you get to the bridge, you may decide not to cross it at all and then all the planning that you did on how to cross the bridge, might just go waste. ***While being Agile, you should only plan and focus on immediate actions required and not on something you may need to do in distant future.*** This methodology has lots of advantages while developing software. It avoids wastage due to avoiding a lot of actions which are not required, in a way that improves the efficiency of development. It enables the stakeholders to monitor and guide the development efforts for the requirements, which are needed by the business.

We develop software for improving some business practices, so whatever helps business practices, should be taken up for development. It usually happens that needs of businesses change, and that too rapidly. That's because it's business; not a process to get some permission or certificate from the government which is unlikely to change for years. *Supporting the changed needs of business in software is Agile development.* And that leads to the success of business. So, being agile is synonymous with being successful. But that also means that whatever the team develops, is not fixed, until the team develops it. Last minute changes are accepted and encouraged so that team develops whatever is needed by the business.

Fixed bid projects

Now let us look at the other angle, that of the fixed-bid projects. In the last 20 years, software development outsourcing has become another way to succeed in your business. It gives you the possibility to get a software developed by some team at a different location in the world, wherever it is profitable to do so. In the beginning, only low-end work like development and testing were outsourced. That did mean that the control was entirely in the hands of the customer. They managed the entire process of development. The team that was developing or testing did not mind this arrangement as they were getting paid. In other words, Macro-Management for doing whatever was told by the controller. If something went wrong and had to be backtracked and created in a different way, good, that meant more hours of work, more money. It meant rework and time-consuming fixes. ***The more the merrier.***

But then two things happened over the years. First, the teams that were only coding, now aspired also to manage the process of development. Second, the customers became wiser. Customers now wanted that if it is something that needs to be reworked or fixed, they did not want to carry the whole burden of it. They wanted to share that with the development team. So now the pattern of outsourcing changed to 'fixed bid'. In a fixed bid project, the customer floats a requirement as a sort of an auction and teams bid for those requirements to be developed into software. One obvious thing about this is that a team cannot arrive at their bid if the requirements are not entirely known. If they do bid for developing a software for which the requirements either are not fully known or are changing from time to time, then the bid will also

[1] Golfers with a handicap of 5 or lower are said to be Category 1 players

keep on changing in line with those changes in requirements, and will not be fixed bid.

Categorically speaking, do customers know the exact requirements of the software they want and do those requirements remain unchanged until the software is completed and accepted? I have come across many projects and product development where the customer had only a high-level idea of what they require from the software. Those requirements do change over time when customer and users get deeper insights into the way that the software is going to work, and that happens only after that software is created up to a particular stage. If the customer knew about the requirements in detail before the software development began, then that project need not be Agile. But knowing full detailed requirements before the development starts, is a myth.

Agile and Fixed Bid Juxtaposition

Being agile and bidding for a fixed bid project are two ideas that are in juxtaposition to each other.

Figure1: Agile and Fixed Bid Juxtaposition

Agile needs us to welcome changes, whereas fixed bid cannot accept changes in requirements. What are the chances that the team going for fixed bid project can also remain an agile team? Very slim. But then as an Agile coach, I do want the benefits of being Agile be available to companies working on fixed bid projects.

Let me recommend a few things that came to my mind and which I tried out in some organizations. It is in the interest of the customer to accept the fact that requirements are not frozen up front, but are kept in a fluid state until they reach a deeper level of understanding of these demands. In this way, the correct and exact requirements will be met and not the ones that are partially understood.

Fluid requirements also mean elastic time and money. Let us have a look at the tradeoff triangle. To adjust to elasticity in requirements, the schedule and cost also need to be flexible. This tradeoff is tough to accept for a customer.

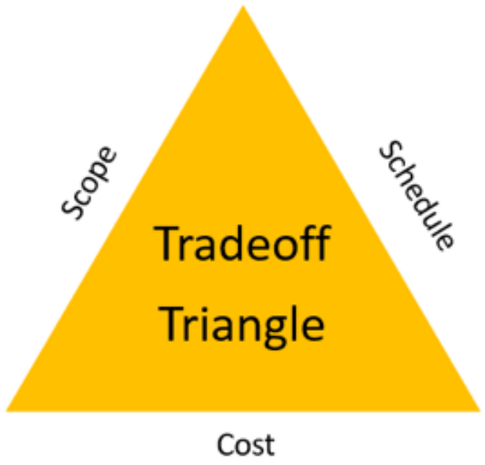


Figure 2: Tradeoff Triangle

Usually, customers who give out contracts to create software with full outsourcing, have a fixed mindset. Customers assume they know what they want and do not want to accept changes in requirements because it affects their budgets. At times, they even have a distrust about the third party that is taking the contract of outsourcing. It is essential to change that mindset in the interest of both the involved parties but this article is not about how to do that.

For this article, I assume that the customer has made up that mindset, and it is almost impossible for us to change that. The real tricky issue is; how do we address such a situation now if we want to remain agile?

Being Agile in Fixed Bid Projects

At the beginning of the project, the feature level expectations of the customer are stated by them. The development team should apply a top-down approach where they know the top level requirements and makes assumptions about details of requirements once they come lower down the level. The development team makes a high-level effort estimate and offers a monetary bid for developing that software. What we understand here is that the team makes a bid with the knowledge of requirements that are known at that time, and we also expect that those requirements may change a little over time, without any significant changes. With these requirements and budgets, a framework is created for the estimation of the number of team members and schedule of completion. Once this framework is in place, now we should think of **periodic delivery**. We can decide the periodicity of delivery over the possible implementation of a block of features. Let us call it by the commonly known term of 'Release.'

When we start development for such a project, we make certain assumptions about the time required for creating that software and depending upon that, the money to be spent on it. We need to work within those parameters and still should have some amount of flexibility to remain Agile. We need to synthesize the waterfall model of development with iterative development model. Waterfall model is natural and conducive for fixed bid project. Iterative development is instinctive for Agile teams. We want the benefits of both, for our process.

As a part of such a synthesized process, my first suggestion is to identify the Macro-Management tasks clearly, as against the Micro-Management tasks. What I mean by Macro-Management is that a part of the product creation is devoted to non-development tasks like getting feedback about the released part of the software, refining next set of requirements, updating design documents and setting budgets with approvals for next release. For Macro-Management tasks, we can adapt a long cycle release process that has many steps in the process.

Micro-Management tasks to be executed by the team of developers every day include requirement tracking, efforts management, code management, build management and deployment management, etc. This part of the release iteration should be further divided in smaller iterations and should be executed by the Agile team. My observation is that the release cycle made up of 3-4 development iteration, works well.

In the early part of the release cycle, macro management is the primary goal. Macro management activities ensure that we remain in the budgeted time and costs. Those are the activities that put the focus on long-term goals, achievable and ensures that we do not go beyond the bid that we have made. Once the activities related to macro management are over, only then the iterative development process starts. See figure 3 on the next page.

In this model, we are not expecting that feature level demands of the customer are going to change much. That is why we can schedule and implement them in a particular release. A number of such releases with time and cost for them, as well as feature level scope is decided up front. At the same time, it also provides flexibility to manage development process for requirements that are part of the feature and address the feedback given by the stakeholders, as is done by Agile teams. This adjustment of detailed level scope gives strength to the Agile teams.

Let us understand the macro-management tasks first and then the micro-management tasks.

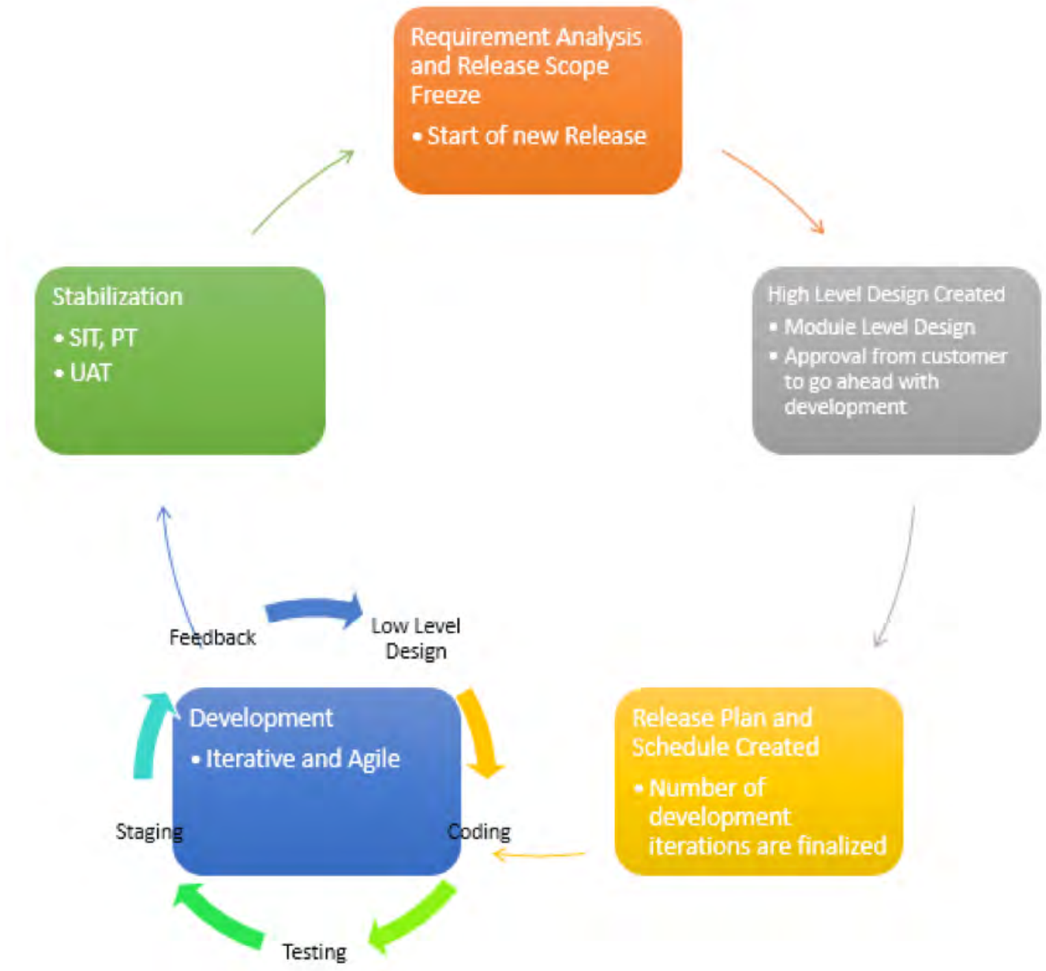


Figure 3: Synthesized model of long release cycles consisting of shorter development iterations

Macro-management tasks

1. **Requirement analysis and release scope freezing:** This is the beginning of the release cycle. The development team and stakeholders define the requirements in as much details as possible. The scope of the release is limited to features that will be implemented. The team should not look at the time to implement those features at this moment, since the release cycle can be of variable length. Factors other than time like the dependencies, risks, etc. should be considered.

2. **High-level design creation:** The design team creates or changes the high-level design diagrams. Usually, it is done at the module level to ensure that communication between components is well defined and is not conflicting.

3. **Plan and schedule the development activities:** A gross level of the estimate is created to complete the development for release. A plan of implementation is carved out with the number of development iterations. These iterations are scheduled.

4. **Development:** This is an iterative activity where each iteration of development completes the part of the release scope. We will discuss the specific actions under this task in the micro-management section.

5. **Stabilization:** During this period, the testing team tests the completed part of the software. It extends the testing that is done by the development team, so that System Integration testing, Performance testing, etc. are carried out. It may also include User Acceptance Testing. During this part of the release cycle, some of the members of the team start working on the first activities of the next release, so that rest of the team can start with second part of the next version as soon as they complete the stabilization task.

Micro-Management Tasks

These micro-management tasks truly define the way an Agile team works.

1. **Identify requirements that are to be taken up in an iteration:** This is also called as Sprint Planning in SCRUM. Depending upon the capacity of the team and the known estimate of efforts for each requirement, the requirements that can be completed in the iteration are identified. Other criteria for selection of requirements can be dependency relationships, risk analysis, value to efforts ration, etc.

2. **Work breakdown:** Each requirement is broken down into tasks that are needed to implement that requirement. Ideally, these tasks should be small enough to be completed in 8 hours. Each of them can be assigned to a different person if necessary.

3. **Development:** Coding, Unit Testing, Manual Testing, etc.

4. **Staging:** Deploy the created parts of the software on testing or staging server from where they can be executed and demonstrated to stakeholders.

5. **Feedback:** Seek and accept the feedback from stakeholders so that any corrections and changes can be incorporated.

Handle Budget Overruns

By taking a top-down approach and implementing planned multiple releases of software, we are reducing the chances of the project going above the budgeted limits. But in spite of this, we should also address the important aspect of what happens if we cannot complete the development in the budgeted cost and are likely to exceed the bid amount. Off course, we do not have the freedom to adjust features, requirements and schedule to suit the monetary constraints.

Let us understand why cost and time overruns would happen in spite of top-down iterative approach that we have taken. ***Scope creep*** is the foremost reason for this budget overshoot. Since the budget is fixed, we should take care of the following points right from the beginning of the project to avoid such overruns:

1. Convince the customer that implementation of all necessary parts of the software is much more important than strict adherence to the bid amount. Some flexibility should be built into the contract.

2. Prioritize the requirements in such a way that only the lowest priority requirements remain when the budget overshoot happens. You can stop further development keeping the customer informed about remaining low priority requirements.
3. Focus and freeze the releases in advance so that budget overshoot does not happen.

4. Ensure that there is no scope creep at the level

of features.

5. Ensure that number of actual releases do not exceed the planned number of releases.

6. Keep the customer informed about the scope creep if it happens and then negotiate for adjustment in scope to ensure that number of releases do not exceed the planned releases.

Conclusion

It is not natural for a team doing contractual fixed bid development to become agile. If such a team desires to become agile, it will have to synthesize the waterfall model of development with an iterative process. In this article, we have discussed an option for meeting these two ends. **Long cycle planned releases with short iterations of development is the suggested option**. By planning the releases, we are ensuring the adherence to budgeted time and cost which are in line with the bid but at the same time, we are giving the liberty of iteration planning and requirement level implementation freedom to the development team for ensuring agility ■

About the Author

subodh sohoni

Microsoft® MVP Most Valuable Professional

Subodh *is a consultant and corporate trainer. He has overall 28+ years of experience. His specialization is Application Lifecycle Management and Team Foundation Server. He is Microsoft MVP – VS ALM, MCSD – ALM and MCT. He has conducted more than 300 corporate trainings and consulting assignments. He is also a Professional SCRUM Master. He guides teams to become Agile and implement SCRUM. Subodh is authorized by Microsoft to do ALM Assessments on behalf of Microsoft. Follow him on twitter @ subodhsohoni*

Yacoub Massad

Object Composition With SOLID

Introduction

By applying the SOLID principles, the Single Responsibility Principle (SPR) in particular, our code base becomes a large set of small classes. Each of these classes is responsible for doing one little thing. As the result of the small individual things that these classes do, we get a large and complex application behavior by composing instances of these classes together.

These small classes have to be highly composable to make it easy for us to compose them together to get the complex behavior that we want for the application. Classes become composable by having their dependencies injected into them instead of them creating these dependencies; and by having dependencies on abstract types or interfaces, instead of concrete classes. This behavior is what Dependency Injection (DI) is all about.

For the purpose of demonstrating these ideas, I have created an example C# application on GitHub. I am going to discuss the changes I made to that application to modify its behavior, paying particular attention to the role of object composition in this process of behavior modification.

Unlike the other articles that I have published here at the DNC magazine, you will have to **download** and at least examine the application code if you want to benefit the most from this article.

About the example application

The main feature of the application, the Document Indexer application, is to read text documents from the file system, extract the list of distinct words for each document and store the documents along with their extracted words into the database. Such information can be used later by another application to search documents quickly .

Although the main feature of the application is simple, we are going to extend the application to add more features. More specifically, we are going to add features to detect already processed documents, record performance information, track the number of processed documents, move processed documents to another folder, retry database access in case of errors, record database access errors to the Event Log, and finally run the whole process continuously.

Please note that in this application, I am not going to care about many of the software development practices, e.g. testing, project structure, and data access patterns. I am also not going to care much about the performance of the application as well. **The primary focus of this example is to show object composition in action.**

The URL for the GitHub repository where the example resides is https://github.com/ymassad/CompositionExample

The initial build

You can browse to the initial build using the following URL: https://github.com/ymassad/CompositionExample/tree/InitialApplication

In order to run the application, you need to do the following:

1) Make sure that you have SQL Server installed (any version will suffice)

2) Copy the Documents folder that you find in the repository to any place on your file system. This folder contains 100 sample documents.

3) Open the solution file from Visual Studio. I have used Visual Studio 2015 community edition to create the solution, but it will also work with Visual Studio 2013.

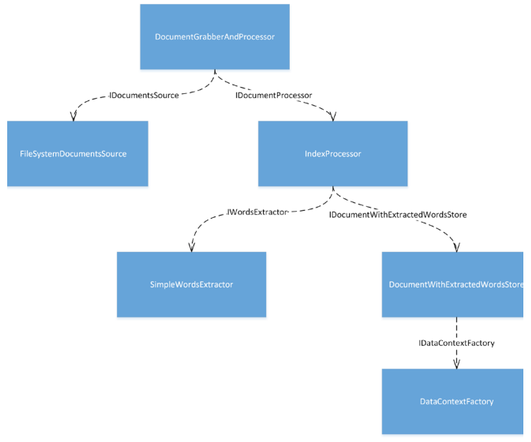
4) Open Settings.Xml, change the *ConnectionString* element value to match the settings of your SQL server.

5) In Settings.Xml, also change the *FolderPath* element value to match the path where you stored the documents.

6) Build the solution and run the program.

Go ahead, download the application, and take a look at the code. Most importantly take a look at the Composition Root in Program.cs.

Here is the object graph that is created in the Composition Root:



The `DocumentGrabberAndProcessor` object is the root object, and it implements the `IRunnable` interface so that it can be triggered to do its job. It invokes its `IDocumentsSource` dependency to obtain documents and passes them to its `IDocumentProcessor` dependency to process these documents.

Currently, we have a `FileSystemDocumentsSource` that gets documents from the file system. For `IDocumentProcessor`, we have an `IndexProcessor` class that extracts words from the document via its `IWordsExtractor` dependency and then uses its `IDocumentWithExtractedWordsStore` dependency to store documents along their extracted words.

Having these interfaces is the way that we make the system modifiable. These interfaces are our extension points. In the next section, we will use such extension points to add new functionality to the system. Now let's run the application.
After running the application, a new database will be created in SQL Server called DocumentIndexer (unless you changed its name in the connection string). You will find two tables; the Documents table and the IndexEntries table.

Each row in the Documents table will contain the

document filename and the document content. For each distinct word in each document, you will find a row in the IndexEntries table that specifies the word, and that refers to the corresponding row in the Documents table.

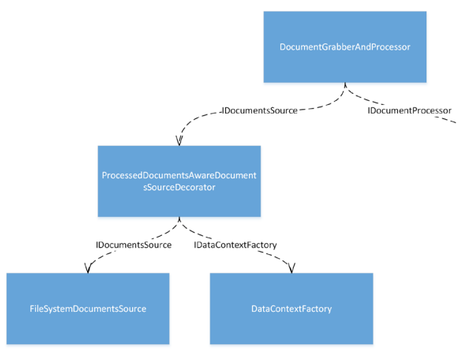
A feature request:
We need the program to be aware of the documents it processed already

As it currently works, the program will always take all the documents, process them, and store the data in the database, even if documents are already processed before.

We need to change the behavior of the system to check if documents are already processed before we process them. What is the seam that allows us to inject such new behavior?

I see two options; we could create a decorator for `IDocumentsSource` that filters the processed documents out, or we could create an `IDocumentProcessor` that skips a document if it is already processed. Let's go with option 1.

We will create a decorator for `IDocumentsSource` that communicates with the database, and filters out the already processed documents before returning the results to the consumer. We will call it `ProcessedDocumentsAwareDocumentsSource`. Here is how the relevant part of the object graph looks now:



Go ahead and see how the code looks now. There URL for this commit is: https://github.com/ymassad/CompositionExample/tree/ProcessedDocumentsAware

A note about performance

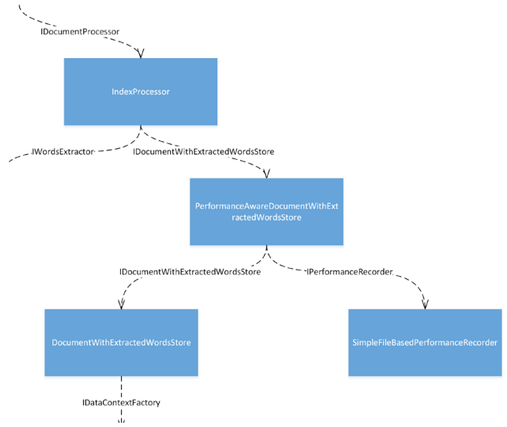
You might have noticed that the solution that we used above is bad regarding performance. One solution is to refactor the `FileSystemDocumentsSource` to split the responsibility of discovering the files (getting file names) from the responsibility of reading the files (getting content). The responsibility of discovering the files would be moved to another interface that we can decorate later to filter files that are already processed.

In this article, I am focusing on showing the power of object composition, so I don't want to do a lot of refactoring. Later in this article, we will add a feature to move processed documents to another folder which will fix this problem.

A feature request: We need to monitor the time it takes to save a document to the database

In this feature, we need to monitor calls to store documents in the database, measure the time, and record it somewhere. For now, we will simply record it to a text file.

We will create a decorator for `IDocumentWithExtractedWordsStore` that measures the time it takes to store the document (by invoking the decorated object) and then invokes an `IPerformanceRecorder` dependency to record the time. For now, we are going to create an implementation of IPerformanceRecorder that records the time to a new line in a text file. Here is how the relevant part of the object graph looks like now:



Go ahead and view the code for the new commit at https://github.com/ymassad/CompositionExample/tree/RecordPerformanceToFile

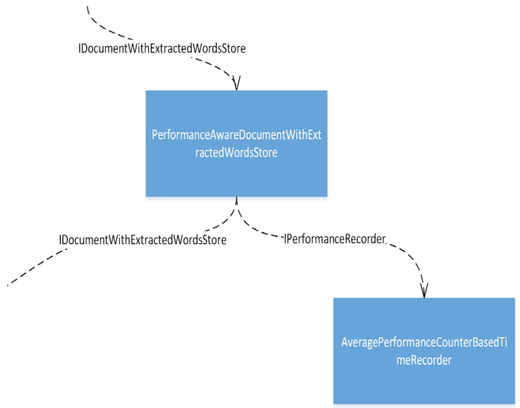
Please note that I have added the location of the text file as a setting in Settings.XML. You might want to change the value of this setting before running the application.

Update: Record the time to a Performance Counter instead of a text file

The product owner tells us that it is not acceptable to record the time to a text file because it is hard to manage. So we decided to use Performance Counters to record the time. More specifically, we decided to record the average time it takes to save a document to the database.

Performance Counters in the Windows operating system are objects that we can use to record performance data. Although you can understand the changes to the code without understanding Performance Counters in details, you might want to read a bit about performance counters and how to use them from .NET before you continue. The following is a link to an article that can help you with that: https://support.microsoft.com/en-us/kb/316365

We currently have an `IPerformanceRecorder` interface. All we need to do is create a new implementation of such an interface that records to some Performance Counter, and then inject an instance of the new class into the appropriate place. Here is how the relevant part of the object graph looks like after the change:



We simply created the `AveragePerformanceCounterBasedTimeRecorder` class and injected an instance in the place where we used to have a `SimpleFileBasedPerformanceRecorder` instance.

Go ahead a take a look at the code at: https://github.com/ymassad/CompositionExample/tree/RecordPerformanceToCounter

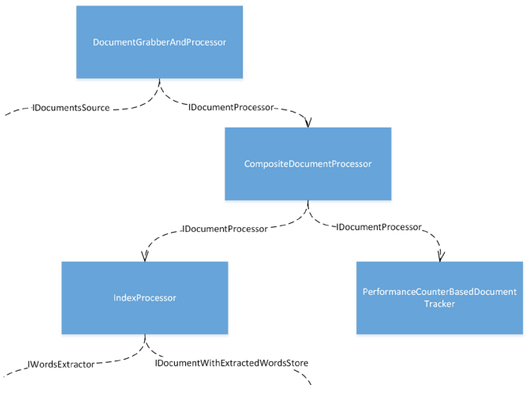
Please note that I have created a method called `EnsurePerformanceCountersAreCreated` that I call at application startup. This method creates the Performance Counters at the operating system level. Usually, such code should exist in installation wizards or deployment scripts. The only reason I put it here is to make it easy for you to run the example.

Please also note that you have to be an administrator or part of the Performance Monitor Users group to be able to access (create and use) Performance Counters.

A feature request:
We need to record to a Performance Counter the number of documents that have been processed

We now need to keep track of the number of documents we process. We decide to track it via a Performance Counter. Where to inject this new behavior?

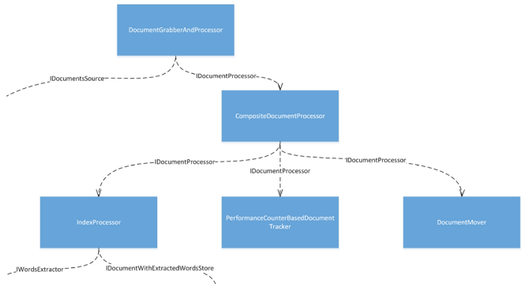
It seems that we need to change the object graph somewhere around the `IDocumentProcessor` interface. We can use the composite pattern to create a document processor that invokes two document processors (or more). We can use it to invoke the current document processor, i.e., the IndexProcessor and a new processor that just increments a Performance Counter. Here is how the relevant part of the object graph looks like now:



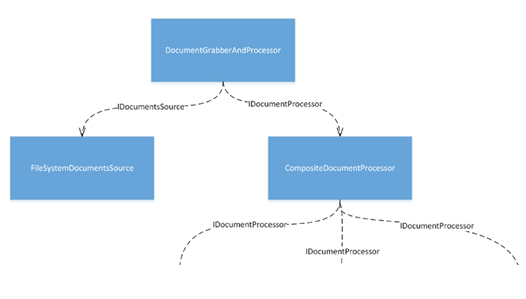
Go ahead and see how the code looks like now: https://github.com/ymassad/CompositionExample/tree/TrackNumberOfDocuments

A feature request: We need to move processed documents to a new folder

After a document is done processing, we need to move it to a new folder. To do so, we are simply going to create a new implementation of `IDocumentProcessor` (namely the DocumentMover) that moves the document to some folder. We then inject it as a third document processor in the `CompositeDocumentProcessor` instance. Here is how the relevant part of the object graph looks like after the change:



Now, since we are moving the processed files into a different folder, we no longer need to check if the files in the source folder are already processed. Therefore, we remove the `ProcessedDocumentsAwareDocumentsSource` instance from the object graph. Here is how the relevant part of the graph looks like:



Here is the link for this commit: https://github.com/ymassad/CompositionExample/tree/MoveProcessedDocuments

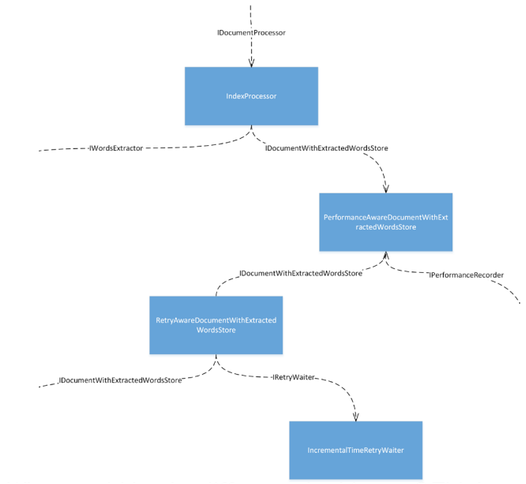
A feature request: We need to retry storing documents to the database in case of a database error

Sometimes when we try to store a document, the operation fails and we get an exception. Sometimes the reason of failure is transient. What we would like to do is to retry to store the document in case of an error. We could, for example, retry the operation five times before giving up.

We are going to create a decorator for `IDocumentWithExtractedWordsStore` that retries the invocation of the method on the decorated store. We will call this decorator `RetryAwareDocumentWithExtractedWordsStore` for now. We will make it possible to inject different retry strategies by making this class depend on an `IRetryWaiter` interface. The current implementation of IRetryWaiter, the `IncrementalTimeRetryWaiter`, increments the wait time by a constant amount in every retry.

Here is how the relevant part of the object graph looks like now:

See how we injected an instance of the new class on top of the PerformanceAwareDocumentWithExtractedWordsStore object (as a decorator for it). We could have done it the other way around. Consider the following potential change in the relevant part of the object graph:



What would be the difference in this case? Think about it for a minute.

The main difference is that in the first case, the recorded time (to the Performance Counter) would be for a single successful attempt to store the document into the database (which is what we want). In the second case, the recorded time is the total time consumed to store the document successfully into the database including the time for possible unsuccessful attempts.

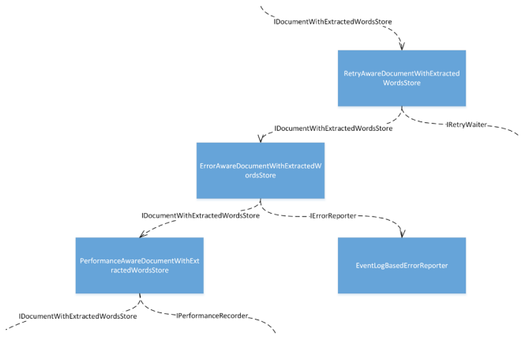
This is an example of how only composing the objects in a different way causes the system to behave differently.

The link for the new commit is: https://github.com/ymassad/CompositionExample/tree/RetryStoreDocuments

A feature request: We need to log to the event log any errors that occur during storing the document in the database

Retrying the operation is not enough, we need to log the error to the event log.
Easy! We need another decorator for `IDocumentWithExtractedWordsStore`.

We create the `ErrorAwareDocumentWithExtractedWordsStore` class and inject it into the appropriate place. Here is how the object graph looks like now:



Notice that the `ErrorAwareDocumentWithExtractedWordsStore` class has a dependency on `IErrorReporter`. This allows us to vary how we report the error later. For now, we provide an implementation of `IErrorReporter` that writes to an Event Log.

Here is the link for the commit: https://github.com/ymassad/CompositionExample/tree/RecordErrors

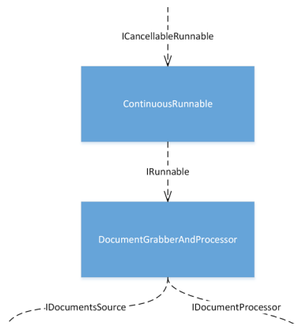
Please note that the code will create an event log source named DocumentIndexer. You need to have administrative privileges for this to work.

A feature request: We need the program to keep pulling new documents as long as it is running

Currently, the program will pull documents once from the folder, process them, and then exit. What we would like to do is make it keep pulling new documents (that users might drop into the input folder) until the user chooses to quit the application.

We create a new interface named `ICancellableRunnable` that allows an operation to run but also supports cancelling that operation by taking a CancellationToken. We then create a `ContinuousRunnable` class that keeps invoking an `IRunnable` dependency as long as it is not asked to cancel.

Here is how the relevant part of the object graph looks like after we create these types:



Here is the link for this commit: https://github.com/ymassad/CompositionExample/tree/ContinuouslyProcessDocuments

Summary:

When we write SOLID code, we create a lot of small classes that each has a single responsibility. We create an application that has a complex behavior by composing these small classes in a specific way. Classes are highly composable because they depend on interfaces instead of concrete types.
This article provided an example of how an application can evolve and highlighted the role of object composition in the process. The code for this example is provided on GitHub so that readers can examine it in detail ■

About the Author

Yacoub Massad

Yacoub Massad is a software developer who works mainly with Microsoft technologies. Currently, he works at Zeva International where he uses C#, .NET, and other technologies to create eDiscovery solutions. He is interested in learning and writing about software design principles that aim at creating maintainable software. You can view his blog posts at criticalsoftwareblog.com.

Shoban Kumar

FACEBOOK SKYPEBOT Using Microsoft Cognitive Services

2016 seems to be the year of bots and the race is getting more interesting. After Microsoft Bot Framework, the latest entrant is Facebook's Messenger platform. This opens up a lot of opportunities for developers. It's much easier to build a complete intelligent Bot with very less effort by making use of all these services.

Here is an idea. An Intelligent Facebook bot that acts as the first point of customer support for a Facebook Page (owned by Brands, Stores etc). You can use LUIS (Language Understanding Intelligent Service) by Microsoft Cognitive Services and build a bot using Bot Framework and connect to Facebook Messenger and Facebook Page.

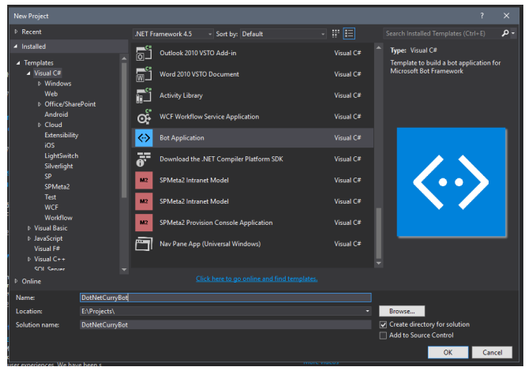
In my previous article Simple Intelligent Bot using Microsoft Bot Framework & Cognitive Services, I showed you how quickly you can build a Bot using Microsoft Bot Framework and make it smart using Cognitive Services. In this article, we will build a better bot and connect it to Facebook and Skype. Let's get started!

Knowledge Guru

Meet Knowledge Guru, a simple Intelligent Bot that can be used to retrieve rich Academic Knowledge. You can ask Knowledge Guru about any Academic topic and retrieve papers published by various Authors and Affiliations within seconds. This dataset is much cleaner than a normal Google search so our Bot is very useful for Students, Researchers and anyone interested in learning.

Our bot will talk to Microsoft Cognitive Services and specifically Academic Knowledge API to retrieve rich academic knowledge as shown below.

Download and install the Bot Application template. Save the zip file to your Visual Studio 2015 templates directory which is traditionally in "%USERPROFILE%\Documents\Visual Studio 2015\Templates\ProjectTemplates\Visual C#". Now open Visual Studio 2015 and create a new Bot Application using the Bot template.
Visual Studio and you will find the Bot template.



Add a new class file named Conversation.cs. Replace the class name in the generated file with the following code:

```
[Serializable]
public class ConversationDialog :
IDialog
```

Add **Microsoft.Bot.builder** Nuget package to the project. Bot Builder framework helps us build more guided conversations. Conversation with Knowledge Guru is a two-step process. One for searching (Interpret) based on the search text, and another one for fetching (Evaluate) publications, links etc based on the selected search result. The Bot builder framework makes it easy to build this guided conversation.

```
public async Task MessageReceivedAsync
(IDialogContext context,
IAwaitable<Message> argument)
{
 var message = await argument;
 string name = string.Empty;
 name = message.
 GetBotPerUserInConversationData<string>
 ("name");
  if (string.IsNullOrEmpty(name))
  {
   PromptDialog.Text(context,
   AfterResetAsync, "Hi! What is your
   name?");
```

```
}
else
{
 bool welcome = false;
 context.PerUserInConversationData.
 TryGetValue<bool>("welcome",
 out welcome);
 if (!welcome)
 {
  await context.PostAsync($"Welcome
  back {name}");
  context.PerUserInConversationData.
  SetValue<bool>("welcome", true);
  context.Wait(MessageReceivedAsync);
 }
 else if ((message.Text.ToLower().
 Equals("hi")) || (message.Text.
 ToLower().Equals("hello")))
 {
  await context.PostAsync("Hello
  again!");
  context.Wait(MessageReceivedAsync);
 }
 else
 {
  //Interpret query
  AcademicResult resp = await
  Utilities.Interpret(message.
  Text.ToLower());
if (resp.interpretations.Count == 0)
{
 resp = await Utilities.
 Interpret(message.Text.ToLower());
}

XmlDocument doc = new XmlDocument();
string xPath = "rule/attr";
string responseMessage = string.
Empty;
string parse = string.Empty;

if (resp.interpretations.Count == 0)
{
 await context.PostAsync("Sorry
 i couldn't find anything. Please try
 again.");
 context.Wait(MessageReceivedAsync);
}
else
{
 context.PerUserInConversationData.
 SetValue<AcademicResult>("result",
 resp);
 int counter = 1;
 List<int> options = new
 List<int>();

 //Get proper text for each
 interpretations
foreach (var interp in resp.
interpretations)
{
 //Add to options
 options.Add(counter);

 doc.LoadXml(interp.parse);
 var nodes = doc.
 SelectNodes(xPath);
 parse = string.Empty;
 List<Item> attributes = new
 List<Item>();

 foreach (XmlNode node in nodes)
 {
  if (node.Attributes.Count == 2)
  {
   if (node.Attributes[1].Name ==
   "canonical")
   {
    attributes.Add(new Item {
    Attribute = node.Attributes[0].
    Value, Value = node.
    Attributes[1].Value });
   }
   else
   {
    attributes.Add(new Item {
    Attribute = node.Attributes[0].
    Value, Value = node.
    Attributes[1].Value });
   }
  }
  else
  {
   attributes.Add(new Item {
   Attribute = node.Attributes[0].
   Value, Value = node.InnerText
  });
 }
 }
 parse = "Papers " +
 ProcessAttribute(attributes);

 responseMessage += $"- {counter} :
 {parse} \r\n";
 counter += 1;
}

options.Add(counter);
responseMessage += $"- {counter} :
Search something else \r\n";

//Post reply
responseMessage = "Here is what I
found. Simply reply with the number
of your choice \r\n" +
responseMessage;
PromptDialog.Number(context,
    AfterChosenAsync, responseMessage,
    "Sorry! I did not understand. Please
    choose from above options.");
  }
 }
}
```

Dialogs are an easy way to present the user with a set of options (just like how Interactive Voice Response IVR works) and accept only those options as response messages. This approach makes error handling easy. In the above code, after the initial greeting is over, the user is presented with the set of options. These options are nothing but search results from Cognitive Services. More detailed results are fetched based on the response from this Dialog. Here are some additional details:

1. In lines 5 and 6, we check if "name" is *null* or *empty* and if it is, prompt a question asking for the name (line 8). This variable is *null* when the conversation starts. **AfterResetAsync** method is called when the user replies with his/her name. This method saves this value in the "name" variable.

2. If "name" variable has a value, then we check the "welcome" variable to greet returning user and new user differently (lines 13-25).

3. We make a request to Academic Knowledge API to get a list of interpretations using `Utilities.Interpret` method (line 29). This method makes an HTTP web request to the Interpret REST Api in Academic Knowledge API. The response is then converted to a custom object `AcademicResult`.

4. The result from above request is saved in conversation variable named "result". This is later used to respond with more detailed answers (line 48).

5. Based on the number of interpretations, we build a menu with an extra option to "search something else" to start a new search (lines 50-93).

6. Prompt a new dialog with a menu using `PromptDialog.Number` which will accept only numbers as inputs from user.

Add the following method to Conversation.cs file.

```
private string
ProcessAttribute(List<Item> attributes)
{
 var atts = attributes.GroupBy(
 g => g.Attribute).Select(grp =>
 grp.ToList()).ToList();
 string attributeTitle = string.Empty;
 string Title = string.Empty;

 foreach (var item in atts)
 {
  string attributeName = item[0].
  Attribute;

  switch (attributeName)
  {
   case "academic#F.FN":
    Title = "in field ";
    break;
   case "academic#Ti":
    Title = "with title ";
    break;
   case "academic#Y":
    Title = "in year ";
    break;
   case "academic#D":
    Title = "date ";
    break;
  case "academic#CC":
   Title = "with citation count ";
   break;
  case "academic#AA.AuN":
   Title = "by author ";
   break;
  case "academic#AA.AuId":
   Title = "with author id ";
   break;
  case "academic#AA.AfN":
   Title = "with author affiliation ";
   break;
  case "academic#AA.AfId":
   Title = "with affiliation id ";
   break;
  case "academic#F.FId":
   Title = "with field id ";
   break;
  case "academic#J.JN":
   Title = "with journal name ";
   break;
  case "academic#J.Id":
   Title = "with journal id ";
   break;
  case "academic#C.CN":
   Title = "with conference name ";
   break;
  case "academic#C.Id":
```

```
 Title = "with conference id ";
 break;
case "academic#RId":
 Title = "with reference id ";
 break;
case "academic#W":
 Title = "with words ";
 break;
case "academic#E":
 Title = "";
 break;
default:
 Title = "";
 break;
}
attributeTitle += Title;
int counter = 0;
foreach (var i in item)
{
 if (counter == 0)
 {
  attributeTitle += $"**{i.Value}**
  ";
 }
 else if (counter == item.Count - 1)
 {
  attributeTitle += $"and **{i.
  Value}** ";
 }
    else
    {
     attributeTitle += $",**{i.Value}**
     ";
    }
    counter++;
   }
 }
 return attributeTitle;
}
```

ProcessAttribute is a helper method which translates attributes, in response from Academic Knowledge API, to words. Cognitive Services API gives us the ability to use complex queries, using Entity Attributes, to search for publications. This method translates an Entity Attribute to an English word. For example, the Attribute Ti is translated to Title. See Entity Attributes documentation page for the complete list of Attributes and their meaning.

Add the following code to Conversation.cs

```
public async Task
AfterResetAsync(IDialogContext context,
IAwaitable<string> argument)
{
 var name = await argument;
 context.PerUserInConversationData.
 SetValue<string>("name", name);
 context.PerUserInConversationData.
 SetValue<bool>("welcome", true);
 await context.PostAsync($"Hi! {name},
 I am Academic Knowledge guru. You can
 simply type in the title to search
 or search by Authors by replying with
 \"papers by AUTHOR NAME\".");
 context.Wait(MessageReceivedAsync);
}
```

The above method saves the name of the user and responds with a welcome message.

Add the following method to Conversation.cs

```
public async Task
AfterChosenAsync(IDialogContext context,
IAwaitable<long> argument)
{
 var choice = await argument;
 AcademicResult result = new
 AcademicResult();

 context.PerUserInConversationData.
 TryGetValue<AcademicResult>("result",
 out result);

string responseMessage = string.Empty;
string linkMsg = string.Empty;

if (result != null)
{
 if (choice > result.interpretations.
 Count())
 {
  await context.PostAsync("Okay! ask
  me.");
 }
 else
 {
  EvaluateResult resp = await
  Utilities.Evaluate(result.
  interpretations[Convert.
  ToInt16(choice) - 1].rules[0].output.
  value);

  int counter = 1;
  foreach (var en in resp.entities)
  {
   EX ex = JsonConvert.
   DeserializeObject<EX>(en.E);
   linkMsg = string.Empty;

   foreach (var link in ex.S)
   {
    switch (link.Ty)
    {
     case 1:
      linkMsg += $" [HTML]({link.U})";
      break;
     case 2:
      linkMsg += $" [TEXT]({link.U})";
      break;
     case 3:
      linkMsg += $" [PDF]({link.U})";
      break;
     case 4:
      linkMsg += $" [DOC]({link.U})";
      break;
     case 5:
      linkMsg += $" [PPT]({link.U})";
      break;
     case 6:
      linkMsg += $" [XLS]({link.U})";
      break;
     case 7:
      linkMsg += $" [PS]({link.U})";
      break;
     default:
      linkMsg += $" [LINK]({link.U})";
      break;
    }
   }
   responseMessage += $"- {counter} .
   {ex.DN} {linkMsg} \r\n";
    counter++;
   }
   await context.
   PostAsync(responseMessage);
  }
 }
 context.Wait(MessageReceivedAsync);
}
```

Above method gets called when the user responds with a paper number of his choice. The result object stored in the conversation variable "result" is used to retrieve the selected interpretation and `Utilities.Evaluate` is used to retrieve more details using Evaluate Method . Results are then parsed and responded with links.

Replace the auto-generated Post method in MessageController.cs file with the following code:

```
if (message.Type == "Message")
{
 try
 {
  return await Conversation.
  SendAsync(message, () => new
  ConversationDialog());
 }
 catch (Exception ex)
 {
  return message.
  CreateReplyMessage("Sorry! Something
  went wrong.");
 }
}
else
{
 return HandleSystemMessage(message);
}
```

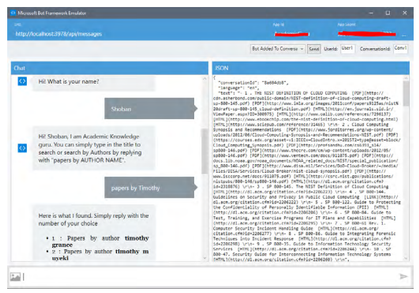
In the above code, we route every message in the conversation to the custom conversation Dialog.

Add the **Utilities.cs** class file attached with the source code to your project. This file has methods to make requests to Interpret and Evaluate API from Academic Knowledge API. This file also contains classes that are used to parse the response easily.

Make sure you get a new subscription key by visiting https://www.microsoft.com/cognitive-services/en-US/subscriptions and subscribing to "Academic Preview" service.

Bot Framework Emulator

Bot Framework Emulator lets you test your bot without publishing online. You can also use this tool to debug the responses that are sent in JSON format. You can get a new App Id and Secret by visiting https://dev.botframework.com/bots/new and registering a new Bot. As of now, you can also use the Bot in the emulator without registering.



The Bot Framework Emulator is good for development testing but I suggest using Azure for a better quality testing. I found that the Emulator is buggy and Message formats were not proper sometimes.

Publish your project to a new Azure App service by right clicking and selecting Publish. Here's a link to the process in case you are not familar with it bitly.com/dncm-publish-azure

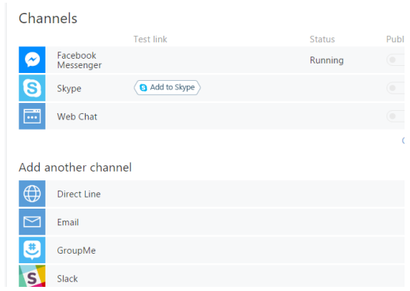
Register your Bot

Visit https://dev.botframework.com/bots/new to edit the Bot and set the Endpoint address to your newly created Azure website in the format **https://{YOUR WEBSITE }/api/messages**

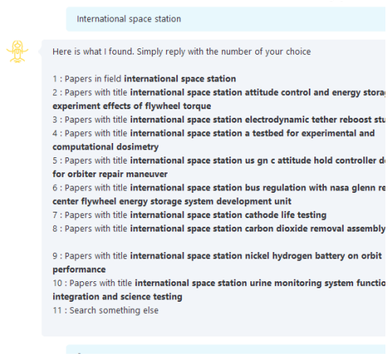
Make sure you update the App Id and Secret in Web.config before publishing your project to Azure website.

Connect to Skype and Facebook

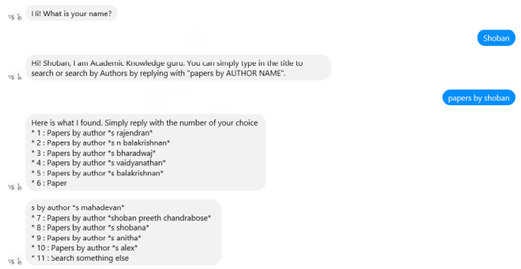
Connecting your Bot to Skype and Facebook requires a developer account in each of these websites. Follow the steps by clicking “Add” in the channel list on your Bot home page.



Here is a screenshot of how the Skype Bot works



Here is how the bot looks with Facebook Messenger



As you can see, Facebook breaks one message into different messages and formatting is not correct. Bot framework is still in Preview and eventually, Microsoft will fix these issues. As a work around, you can check this channel source and respond with different messages for different channels.

Conclusion

The Microsoft Bot Framework and Cognitive Services makes developing intelligent bots, and connecting different platforms, a breeze. But it is still in Preview and not fully matured. Microsoft is constantly adding more features to it and improving it. In this article, we used Microsoft Bot Framework and Cognitive Services to build a bot connected to Facebook Messenger and Facebook Page and retrieve academic knowledge using the Academic Knowledge API ■

Download the entire source code from GitHub at bit.ly/dncm24-facebook-bot

About the Author

shoban kumar

Shoban Kumar *is an ex-Microsoft MVP in SharePoint who currently works as a SharePoint Consultant. You can read more about his projects at http://shobankumar.com. You can also follow him in Twitter @shobankr*

Interview...

Eric Lippert

Eric needs no introduction to the C# folks out there, but for others, here's Eric's Bio in his own words -

Eric Lippert is a developer who works on language tool infrastructure at Facebook. Previously he worked on the Visual Basic, JavaScript, VBScript, and C# compilers at Microsoft, and on static analysis tools at Coverity. He's a frequent contributor to StackOverflow, and writes a blog about having fabulous adventures in coding at ericlippert.com.

Dear Readers, we are thrilled to have Eric Lippert to talk to us once again in our 4th Anniversary edition of the DNC Magazine. To jog your memory, we took Eric's interview in May 2013 which can be found at bitly.com/dncm-may13

1. We heard you recently joined Facebook. Tell us more about your work at Facebook.

I've been at Facebook since the beginning of 2016, and I'm having a marvelous time. Facebook is putting a lot of time and effort into open source developer tools, and there are a lot of interesting problems to solve in this space, particularly at the sort of scale that Facebook needs to work at. Specifically, I'm working on improving the Hack language and its compiler; Hack is a gradually-typed version of PHP with an interesting type system.

2. If you were starting out now in your career, what languages would you learn and why?

The specific details of various languages are just that: details. The important thing at the beginning of a career is to get the conceptual tools in your toolbox that you can use to solve problems. I'd suggest to beginner programmers that they learn at least one OO language – C++, C#, Java, Visual Basic – and at least one functional language – F#, OCaml, Haskell, Scheme, and so on. Once you have the concepts solid, it's not hard to move from language to language.

3. As a developer, what's your greatest success story?

Looking back, the thing I'm proudest of is the "Roslyn" C# compiler architecture. Building a new, industrial-grade compiler starting from a blank page, that's a rare treat in this business. I was very lucky to be a part of that project from the very beginning; I am thrilled that it finally shipped and is successful.

4. How much does the fact that Roslyn is a "compiler as a service" with public API affect its internal structure? How similar is it still to a standard "black box" compiler, as taught in a compiler course at the university?

Two things come to mind. First, the Roslyn API was designed to have a very "functional" flavor while still of course being written in an OO language. Syntax trees, for example, are immutable; you don't ever change a syntax tree, you produce a new syntax tree that is the result of editing an old one. Baking that design choice deep into the compiler architecture had far-reaching implications.

Second, the performance requirements of Roslyn are very different from those of a traditional compiler. Roslyn needs to deal with both the "batch" scenario, where the compiler is handed a huge pile of source code and needs to generate assemblies as quickly as possible, and also the "IDE" scenario where many small edits are being made per minute, and the analysis needed for IntelliSense has to be up-to-date and accurate. A huge amount of work went into optimizing the compiler for these two scenarios; you would typically not see that in a traditional compiler.

5. During Roslyn development, did you encounter any discrepancies between the specification and the existing compiler? If so, how did you handle them: does Roslyn's behavior follow the specification or the original compiler behavior?

Constantly! And resolving every one of them required a judgment call that involved meetings between the developers, designers and testers. The spec is 800 pages long, and there are a lot of places where bugs can creep in.

Typically, what I would do if I found a spec violation was first, produce the simplest possible program whose behavior differed from the specified behavior. It is a lot easier to understand these things with a minimal example.

Next I would think about whether there was a realistic example: are there real-world programs out there whose correct behavior relies on the C# compiler not implementing the specification exactly right? If yes, then that is points towards either changing the spec to match the implementation, or deliberately not fixing the bug and being in violation of the specification forever.

If you want to see examples of that, just get the Roslyn sources and look for the words "deliberate spec violation". I did not want anyone to discover the bug later on and fix it, if we had decided to not fix the bug, so all the ones I worked on I very carefully documented.

6. Asynchronous programming with async/await keywords is probably one of the most prominent features, originally introduced in C# 5.0, but has now found its place in many other languages like JavaScript, Python, Dart etc. This feature was improved in C# 6.0 with support for asynchronous calls in catch and finally blocks. Could this programming model be expanded further, e.g. with support of asynchronous iterators (async Task<IEnumerable<T>> that could both yield and await)?

First off, just to clarify the question: asynchronous waits were introduced to C# in C# 5.0, but there were other languages that had features like this much earlier. The design of the feature in C# 5.0 was heavily influenced by asynchronous workflows in F#, for example.

Could the feature be extended further? Absolutely. There is no logical impediment to that sort of combination of features. Like all features, it's a matter of prioritization; someone has to design, implement, test and document every feature, and that means doing work that is then not spent on other features that might be more useful.

7. Although you are not a part of the C# team anymore, are you following the language design for C# 7.0? How do you like the new features? Which one is your favorite and why?

I am following the process with interest, but not participating much. Between working on languages all day, editing programming books, and writing a blog about programming, I don't have a lot of extra time to participate in the C# design process. I do stick my head in occasionally though, and the design team has been good enough to invite me over for lunch every now and then.

I am very pleased with the proposed slate of features for upcoming versions of C#. We worked so long and so hard on Roslyn to make it a solid foundation for rapidly developing new features, and now it is. The features I am most looking forward to for upcoming versions of C# are those that are inspired by functional languages, like tuples and pattern matching. I've been working in OCaml for a few months now and I have gotten very used to having both at my disposal!

8. Why doesn't C# support converting a lambda expression with a statement body into an expression tree? Is it hard to implement? Or did the C# design team not give much importance to it? Do you know if it will be included in a future version?

Converting lambdas to expression trees was one of my features for C# 3.0, so I'm quite familiar with this one! Extending the feature to statements was not implemented in C# 3.0 because it is not necessary to make LINQ work and we were very short on time for that release. The thing about LINQ is that it required a large number of language features to all work together effectively, so any part of those features that was not necessary was cut.

Again, it comes down to opportunity costs. The feature is not conceptually hard, but it is a lot of work. There are a great many statements in C# and you need to write the logic to transform all of them into expression tree formats. There's a lot of tests to write, a lot of documentation to write, and so on. It's a pretty expensive feature but it doesn't add much power to the language; that effort could probably be better spent elsewhere. If we could get it for free, then sure, it's a nice thing to have, but we can't get it for free. I don't know if there are any plans here; I have not heard of any.

9. Many new and upcoming C# features make functional programming with C# easier. Where do you see this going? How do you foresee the future of functional programming with C#? Amidst this, do you see a future for F# or is it just going to be a laboratory for functional features that eventually show up in C#?

As I said above, I love this trend and I see it continuing. There has been a real renaissance of functional programming over the last couple of decades and it is still going strong. The benefits of programming in a functional style are numerous: code is easier to reason about when there are fewer side effects, code can be parallelized more easily and more safely, and so on.

The hard part will be getting these features into C# while still keeping the "spirit" of the language the same. The first time I saw the "pattern matching" feature proposal for C# I was really impressed by how the designers managed to take a feature usually associated with functional languages like OCaml or Haskell, and make it feel like it was a natural extension to C# rather than something grafted on.

I see a bright future for F# as well. Both C# and F# are hitting sweet spots; C# is attractive to OO developers who can benefit from more functional features, and F# is attractive to developers who like programming primarily in functional style but have to interoperate with a whole world full of libraries designed to work with OO languages. The .NET platform is fundamentally a multi-language strategy; there's lot of room for F# in there.

10. We have seen a massive change in Microsoft's attitude with .NET going Open Source and all. Hypothetically, if this would have been the case when C# was introduced to the world, how would have the language been shaped compared to what it is now?

I think whether the source code is public or private doesn't really affect the design of the language. The factor in Microsoft's astonishing switch to embrace open source that will most affect the design of the language is that the design process itself is now open. Getting ideas out for criticism by the public earlier certainly has costs, but the benefits greatly outweigh them.

It's hard to say how things would have been different had the design process been more open from the beginning. I suspect that it would have been harder to get good discussion on big, complicated, far reaching features, like generic types or LINQ, and easier to get feedback on small "productivity" features that affect day-to-day developer life.

11. You have had a remarkable career. How does one become as good as Eric Lippert? What sort of qualifications (both academically as well as professionally) are required to be as successful as you are?

That's kind of you to say, but I want to deny the premise of the question. I know many, many successful developers both in the tools space and elsewhere that have a completely different background than I do. I know people who have PhDs in type theory and people with no formal CS education at all who are successful in programming language design. Personally, I found that having studied computer science at Waterloo was a great preparation, but that is neither a necessary nor a sufficient condition for having a fun career.

The things I see in common amongst the successful people in this industry who I admire are a drive to learn new things, taking enjoyment in passing knowledge on to others, and willingness to stick to it for a long time.

12. Any new hobbies/personal projects that you have been working on since we last spoke to you? We would love to see a video of you playing the Piano or the Ukulele!

I have been pretty busy learning several new-to-me programming languages since I joined Facebook, and thus I've been neglecting my hobbies. There's a nice piano in the Facebook cafeteria though, and I've started playing again recently. I'll see if I can put together a video!

13. Finally to wrap up, share your favorite C# gems or hidden features with our readers.

I'll leave your readers with a little puzzle that is a favorite of mine; it illustrates that the C# type system is maybe more complicated than you think:

```
public class A<T> {
  public class B : A<int> {
    public void M() { System.Console.
    WriteLine(typeof(T)); }
    public class C : B { }
  }
}

public class P {
  public static void Main() { (new
A<string>.B.C()).M(); }
}
```

What do you think this little program prints out, and why? Now actually run it and see if you were right! If your first guess is wrong, don't feel too bad. I got it wrong the first time I saw it too!

• • • • • • • •

Thank you Eric for your time! It was a pleasure interacting with you.

*The interview questions were curated by **Damir Arh**, **Suprotim Agarwal** and **Yacoub Massad** ■*

Damir Arh

PROJECT CENTENNIAL

UWP Migration Path

for Legacy Desktop Applications

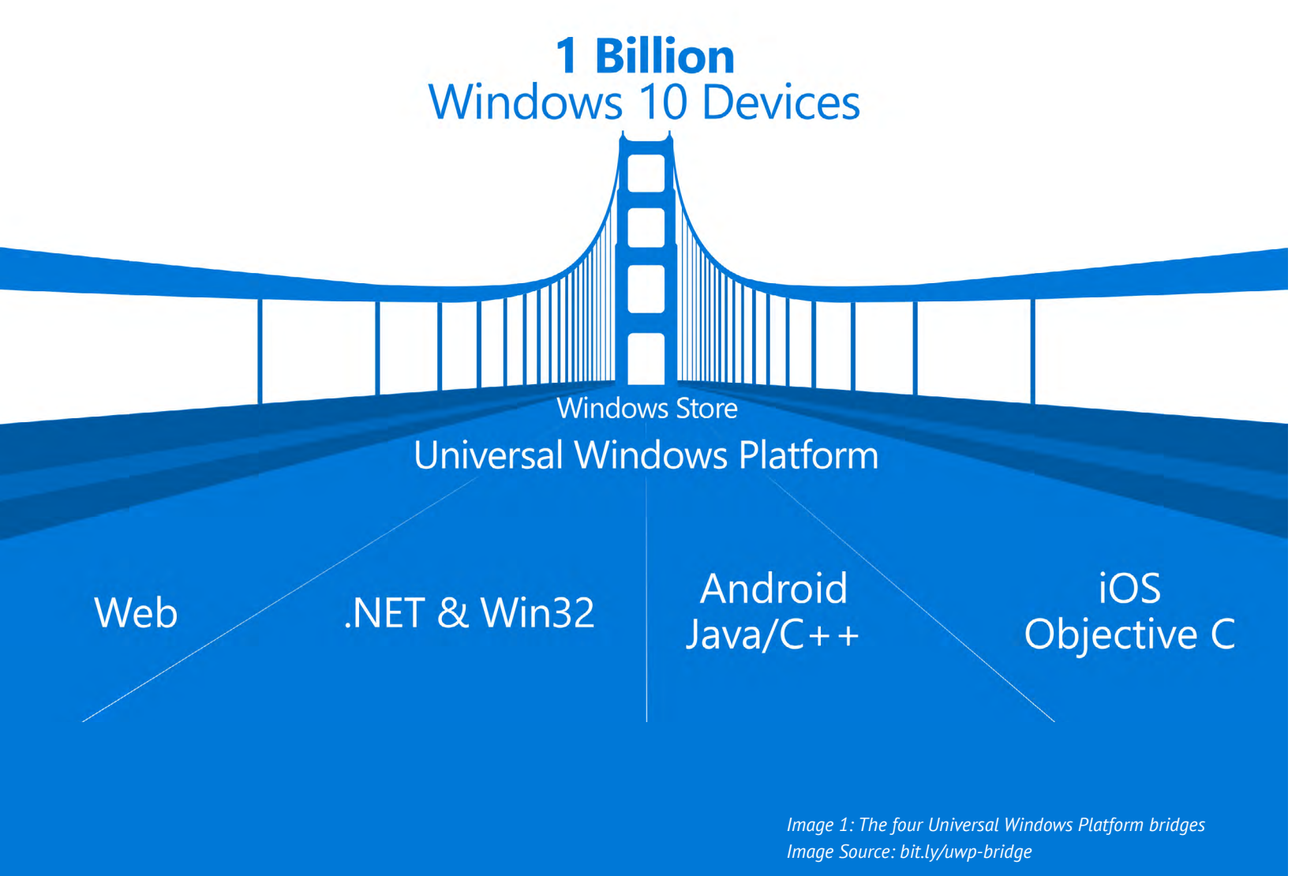


Image 1: The four Universal Windows Platform bridges
Image Source: bit.ly/uwp-bridge

Windows 8 introduced a new type of Windows application in 2011: the Metro style application. Since then, it has been renamed a couple of times from Metro Style apps to Windows Store apps and now it goes by the name of Universal Windows Applications or Windows Store applications. Nevertheless, the essence of these applications has remained unchanged: these apps run in a sandbox with a restricted set of permissions, they are available as packages that by design support full uninstall, and they undergo a standard review process before being included in the centralized Windows Store (an app store for Windows). Up until now, none of this was available to classic desktop applications that existed before Windows 8, making these classic apps second-class citizens.
With Project Centennial, this might change.

Bringing Applications from Other Platforms to Windows

At the Build conference in 2015, Microsoft announced four Universal Windows Platform (UWP) bridges, with the goal of simplifying the conversion of existing applications from other platforms into Universal Windows applications, to distribute them through the Windows Store.

Each of these four bridges has a similar project, dedicated to developing the said feature:

- **Project Westminster**'s goal was to bring existing web applications into the Windows Store.

- **Project Centennial** was all about packaging existing .NET and Win32 desktop apps in such a way so as to include them in the Windows Store.

- **Project Astoria** was aimed at making it easier to port existing Java and C++ applications for Android, to Universal Windows platform.

- **Project Islandwood** was developed to make it easier to port existing Objective-C applications to Universal Windows platform.

Microsoft officially canceled project Astoria in February 2016. The other three projects are now all publicly available, albeit in different stages of development:

- Project Westminster's new name is Hosted web apps. It became a part of Windows SDK with the release of Windows 10 in July 2015. Several published Windows Store applications are already taking advantage of it.

- In August 2015, Microsoft open sourced project Islandwood as Bridge for iOS and released it on GitHub. It is still in active development, with a new preview released almost every week.

- At Build conference 2016, project Centennial finally made it into the preview, as well: Desktop App Converter is now available for download.

Project Centennial – The Bridge to Windows Desktop

Windows desktop applications converted to Universal Windows platform (UWP) get full application identity, just like any other Universal Windows application. Eventually, you can publish them in the Windows Store, with standard application licensing options and full auto-update support. UWP application identity also gives them access to a much larger subset of UWP APIs than an ordinary desktop application. They can create live tiles, register background tasks with any trigger type, create app services, etc.

The conversion process does not make them real universal applications though. They can only run on the desktop device family because they are still using legacy APIs that are only available in the desktop version of Windows 10. Hence, the automatic application conversion is supposed to be only the first step in the process of making them true universal applications that can run on any device type.

To make this transition process easier, an application package with a converted desktop

application can include two separate executables: the original desktop one, and a separate Universal Windows component. Although they are running in two different processes, they can activate each other at the user interface level and communicate bi-directionally using app services (APIs similar to web services that a UWP application can expose). This process should allow a gradual migration of application functionalities from the legacy desktop process to the new UWP process.

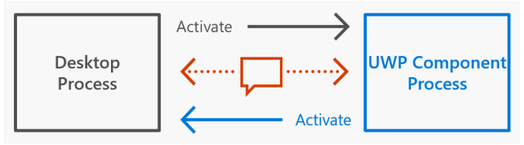


Image 2: Communication between the desktop and the UWP process

Once the migration is complete, the application will, of course, be able to run on all Windows device families. However, even with a desktop executable still in the package, support for non-desktop device families can be declared in the application manifest. When installed on those devices, the application will only run the UWP component of the package. With proper design, such an application can still be fully functional on all devices. It can choose to support only a reduced feature set in comparison to Windows desktop on all the other devices.

A New Way to Distribute Desktop Applications

Desktop App Converter will convert an existing game or desktop application (Win32, Windows Forms, or WPF-based) into a Universal Windows application package (*.appx). Until these packages are allowed into Windows Store, sideloading will be the only way to install them. In Windows 10, Microsoft removed all licensing restrictions for sideloading, making it available to all users. They only need to enable sideloading in Windows settings.

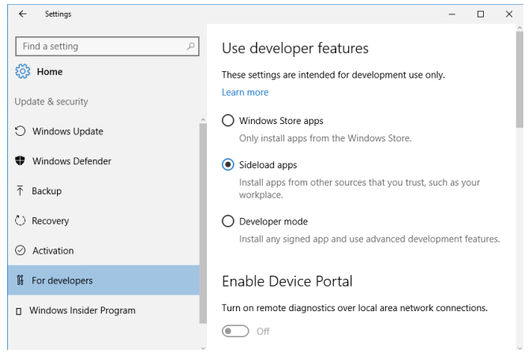


Image 3: Sideloading enabled in Windows Settings

A packaged desktop application gets its own application manifest for defining its capabilities and declarations. Since the desktop process will be running outside the sandbox even after the conversion, the capability restrictions will only apply to the UWP part of the package.

Although the desktop part of the package will keep using Win32 and .NET APIs in full trust mode, its environment will still be much more isolated than that of a standard non-converted desktop application. All its file system and registry access will be virtualized in a manner similar to App-V. Without any modifications to the application source code, the actual file system and registry will remain untouched. Any changes will automatically be written to the package local storage as if the UWP file and settings APIs were used.

This approach makes uninstalling the application a breeze. By only deleting the package local storage folder and the installation directory itself, all traces of the application will be completely removed without the risk of leaving behind large nodes of registry settings. The solution is not perfect, though, and some features will require additional intervention to keep them working. For example, desktop applications register supported file extension by adding entries to registry during installation or from a running application. Due to registry virtualization, this will not work anymore. Instead, a File Type Association declaration will need to be added to the application manifest to achieve the same result.

It is not so easy to resolve every compatibility issue. The limitations of the virtualized environment, in which the packaged desktop applications are running, will break some of them after the conversion. The following are the most notable restrictions:

- There is no kernel mode available; therefore, drivers and Windows services will not work.
- No in-process (shell) extensions can be registered
- No modifications to the application installation directory or the local machine registry hive are allowed.
- Due to isolation, the application cannot share any of its files with other applications using the AppData folder.

Code changes can circumvent some of these restrictions, e.g. files can be shared between multiple applications from the same publisher by using a common publisher cache folder. Others will require a larger redesign of the application to keep it working, or even make it impossible. Such applications will remain restricted to desktop devices and have to keep their current distribution model.

Using the Application Converter

To use Desktop App Converter your computer must fulfill the following requirements:

- It must be running a 64-bit version of Windows 10 Enterprise or Pro edition with Anniversary Update preview installed, i.e. build 14342 or newer.
- It must support hardware-assisted virtualization and second-level address translation (SLAT) and have both enabled.

The tool consists of two downloads: the converter itself and a Windows 10 base image, matching the Windows build you are currently using. You also need to have Windows 10 SDK installed, as the converter uses *MakeAppx.exe* from the SDK to build the final package.

To set up the converter, unblock and unzip its archive, navigate to its folder, and run the following PowerShell commands as administrator:

```
Set-ExecutionPolicy bypass.\
DesktopAppConverter.ps1 -Setup
-BaseImage .\BaseImage-14342.wim
-Verbose
```

The second one requires the new Containers Windows feature. The script will enable it and automatically reboot the computer, if necessary. The setup process will take a significant amount of time. Once it completes, you will be ready to run the converter on a desktop application of your choice with the following command:

The second one requires the new Containers Windows feature. The script will enable it and automatically reboot the computer, if necessary. The setup process will take a significant amount of time. Once it completes, you will be ready to run the converter on a desktop application of your choice with the following command:

```
.\DesktopAppConverter.ps1 -Installer
"<MyAppSetup.exe>" -InstallerArguments
"<SetupArguments>" -Destination
"<DestinationFolder>" -PackageName
<PackageName> -Publisher
"CN=<PublisherName>" -Version <Version>
-MakeAppx -Verbose
```

You need to replace the placeholders as follows:

- *<MyAppSetup.exe>* is the full path to your application setup. Usually, it will be an executable or a Windows installer package, but a batch script will work as well. The only requirement is that the setup can fully execute silently without any user interaction.
- *<SetupArguments>* includes any arguments that need to be passed to your application setup for it to run completely silently.
- *<DestinationFolder>* is a folder on your machine where you want to output the generated package and its contents.
- *<PackageName>* is the name that you want to use for your application package (without the .appx extension).

• *<PublisherName>* is the publisher name that you want to use in your package. You will need a code-signing certificate with a matching common name.
• *<Version>* is a four-part version number (e.g. 1.0.0.0) that you want your package to use.

The converter will use the downloaded Windows base image to create an isolated Windows environment and run the existing setup for the desktop application inside it. It will capture any changes to the registry and file system during the installation procedure, and include them in the generated application package:

• The application files that were copied to the installation directory.
• Any global linked libraries that the application depends on.
• A local registry hive containing all the registry changes.

For some applications, the converter might not be able to detect the installation path or the application executable to run. In this case, the conversion process will fail, and you will need to specify additional arguments for *DesktopAppConverter.ps1:*

```
-AppInstallPath
"<InstallationFolderPath>"
-AppExecutable "<ExecutablePath>"
```

Replace the placeholders as follows:

• *<InstallationFolderPath>* is the absolute path of the folder containing the application files after the silent install is completed.
• *<ExecutablePath>* is the absolute path of the application executable that needs to be launched.

Before you can install the generated application package, you will need to sign it. All the required tools are included in Windows 10 SDK. If you do not have them in the path, you can find them all in `C:\Program Files (x86)\Windows Kits\10\bin\x64` folder. Invoke them in the following order:

```
MakeCert.exe -r -h 0 -n
"CN=<PublisherName>" -eku
1.3.6.1.5.5.7.3.3 -pe -sv <cert.pvk>
<cert.cer>
pvk2pfx.exe -pvk <cert.pvk> -spc <cert.
cer> -pfx <cert.pfx>
signtool.exe sign -f <cert.pfx> -fd
SHA256 -v "<GeneratedPackage>"
```

Placeholder values are as follows:

• <PublisherName> must match the publisher name, which you used when generating the package.

• <cert.pvk> is the filename for the generated certificate private key

• <cert.cer> is the filename for the generated certificate public key.

• <cert.pfx> is the filename for the generated certificate PKCS #12 file.

• <GeneratedPackage> is the full path of the application package file generated by the converter.

Before installing the package, you need to set up trust for the certificate you signed it with. If it was a self-signed certificate, created with the command above, just double-click the generated *<cert.cer>* file and import it as a trusted root certification authority.

Assuming that you have unlocked your copy of Windows 10 with Anniversary Update for sideloading, you can now install the generated and signed application package:

```
Add-AppxPackage "<GeneratedPackage>"
```

Again, replace *<GeneratedPackage>* with the full path to the now signed generated application package. In the final release of Windows 10 Anniversary Update, it should be possible to install the .appx packages with a double click, without invoking any PowerShell cmdlets.

Additional tooling for packaging desktop applications was already announced and is expected to be available before the final version of project Centennial:

• WiX and InstallShield (and maybe other setup tools vendors) will support the direct generation of AppX packages for desktop applications without the conversion process.

• A special project template will be available for Visual Studio, which will support the creation of AppX packages from the desktop and UWP components in the same solution. It will even support debugging of both package components.

Conclusion:

Currently, the focus of project Centennial is on Line of business applications (LOB) for the enterprise. In this environment, sideloading will be the standard method for installing the application packages. Nevertheless, Windows Store support for packaged desktop applications is planned and might even be available at the same time as the final version of Windows 10 Anniversary Update is released.

There have already been announcements that more of the existing Win32 APIs will be available for UWP apps (i.e. in .NET Core). The final goal is to include everything, except the platform specific parts (registry APIs, Windows Forms, and WPF, etc.).

If all this comes true, we can look forward to a new migration path to UWP for existing desktop applications that does not include a full rewrite ■

About the Author

damir arh

Microsoft® Most Valuable Professional

Damir Arh has many years of experience with Microsoft development tools; both in complex enterprise software projects and modern cross-platform mobile applications. In his drive towards better development processes, he is a proponent of test driven development, continuous integration and continuous deployment. He shares his knowledge by speaking at local user groups and conferences, blogging, and answering questions on Stack Overflow. He is an awarded Microsoft MVP for .NET since 2012.

Subodh Sohoni

Developing a non-trivial application that targets multiple platforms like Android, iOS and Windows is not something an individual developer can do easily. It requires collaborative efforts of a team. When a team develops a software, it requires many more services than what a single developer may require. A team requires a centralized source control, version control, server-side build that also takes care of verification tests, and it also requires automated deployment after the build to the staging server, so that the testers can immediately start executing the tests on the deployed build.

Cross Platform Mobile Application Development in a Team using Xamarin, VSTS and Azure App Service

Microsoft offers a platform for such application development that requires a team. This platform in the cloud is called **Microsoft Visual Studio Team Services (VSTS)** and on your premises it is called Microsoft Team Foundation Server (TFS).

In this article, we will take an overview of how a team can use VSTS for all these services for a cross platform mobile application that they are developing on Azure.

Any mobile application can be divided in two tiers. The UI tier is on the mobile phone or a device like tablet. Resources on such a mobile device are limited. Some of the resources like data are at a remote location where a centralized database exists. For doing heavy computing, such a mobile device is not appropriate. Which is why the other tier of the application is located on a server that is accessible to all the devices through the Internet. Such a server based application tier can do heavy lifting of the computing load and can also access centralized resources like data, very easily. Microsoft Azure offers support for this service tier application to be hosted, monitored and managed. This type of application falls under the group of App Services. It is an MVC application which exposes a data table in the form of entity objects.

Let us take a step by step tour of creating such an application and use VSTS to implement Continuous Integration Continuous Deployment (CICD) of it. **The focus of this tour is not to learn how to code the front end of a mobile application** as that can be created even by an individual developer. Rather we are going to focus on the topic of **how a team can develop back end service application for a non-trivial mobile application and then use VSTS to build it and deploy it on Azure**. The last stop of this tour guides about how the front end mobile app that is linked to this back end service app, can be created. This tour requires Visual Studio with Xamarin and Azure SDK 2.9 onwards to be installed on your machine. We also assume that you have an Azure Account (either paid or a free trial account). We begin the tour by logging on to the New Portal of Azure (http://portal.azure.com).

Create back end of mobile app in Azure App Service

Once we are logged in to the Azure portal, we will create a new App Service by clicking on the Add button at the top left corner of the Home page and then select the option of Web + Mobile.

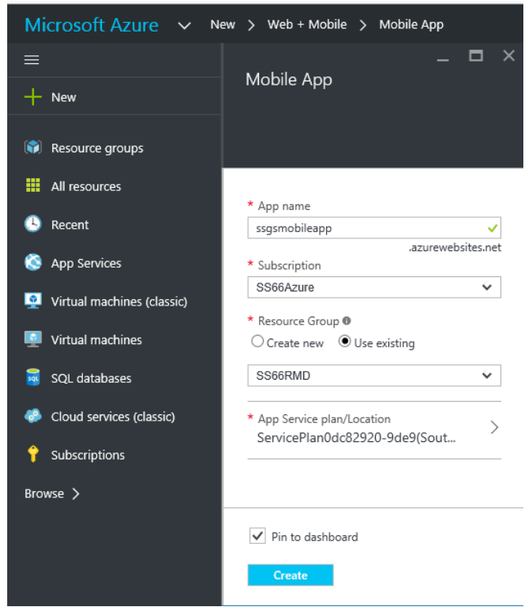


Figure 1: Create Mobile App under Azure App Services

We will give a name to the app, like I have given it as "ssgsmobileapp" and select a resource group if you already have one. We can also create a new resource group. We will then select a App Service Plan with Location. The price of the application is dependent on the selected App Service Plan. There is a shared (free) plan also available which we can select for this overview. Finally we will click on the Create button to create and host the new Mobile Service App.

After the Service App is created, we can use the Quick Start wizard from the Settings blade of that app to configure necessary aspects of the app. First of all, we will select the target platform for the app. In this example, I have selected Xamarin.Forms that will use Xamarin to create common code for iOS,

Android and Windows mobile platforms.

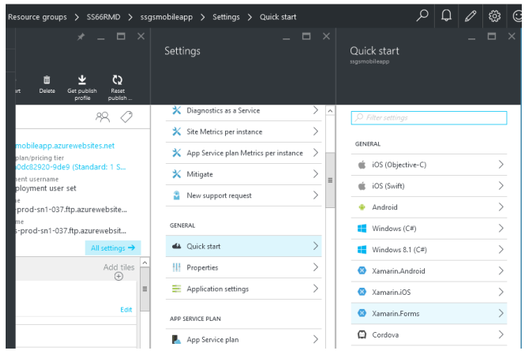


Figure 2: Select App type in the Quick Start Wizard

Create Database and Table API

As a next step, we can configure the database that will be containing the data of our application. If the database does not exist, a new database and its connection string can be created.

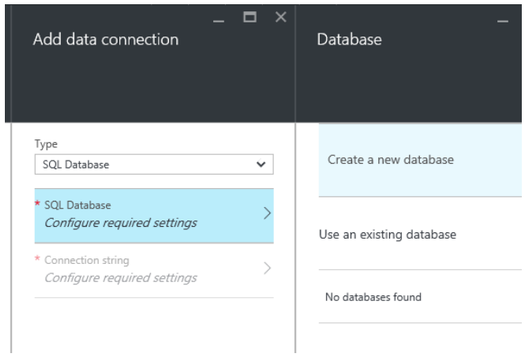


Figure 3: Create database for the app

After the SQL database is created, we move on to create the Table APIs for the database table that we have created. We can select either Node.js or C# as the language for generating those APIs. Here I have selected C#, just because I am more comfortable using C#.

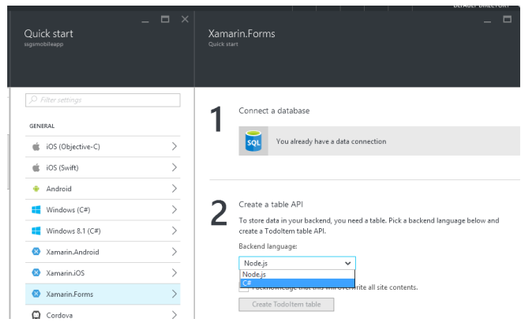


Figure 4: Select Language for table API

Now we can Download the application which will be created and zipped, before it is downloaded on our machine. That downloaded application can be unzipped and copied wherever we want. It has a complete solution which we can open in Visual Studio 2015. Since we are going to use that application with the code that is generated, and not make any changes in that code, we are not showing you the code view here. You can open it in Visual Studio 2015 and go through the code if needed. Ensure that Xamarin is installed on Visual Studio 2015. Until a few days back, Xamarin could be installed on Visual Studio 2013 too, but now that version seems to be retired from Xamarin.com website. You can download Visual Studio with Xamarin over here https://www.visualstudio.com/en-us/features/xamarin-vs.aspx .

Create a Team Project and add project to Source Control

Before we open the project in Visual Studio 2015, we will create a repository in VSTS for the code. This is done by creating a new Team Project on VSTS. I have created a team project named "AzureMobileApp". It is using the TFVC (Centralized) repository but everything works well even if you are using Git. Instead of check-in you do the Commit and Push to trigger the deployment.

Now we can open the solution. As soon as it is opened, it should be built locally so that NuGet packages get restored. After that we will add the solution to the source control under the team project that we have created earlier. Before taking the next step, we can check-in the code to that source control.

Link Azure App Service to VSTS

We will now open the old portal of Microsoft Azure – https://manage.windowsazure.com. After logging in to the portal, we will open the Service Apps section and select the app that we have created. On the dashboard, we can view the link to integrate the app with the source control so that auto deployment can be configured.

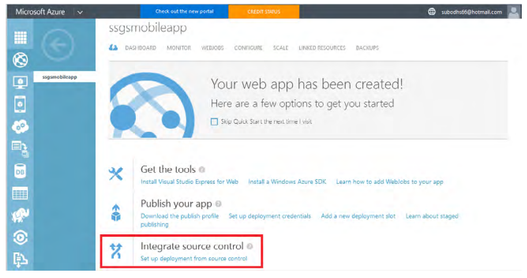


Figure 5: Integrate source control with Service App

We can now select the source control software, which is VSTS, although the old portal still mentions it as Visual Studio Online which is the old name of VSTS. Then you can select the team project name that we have created for this example.

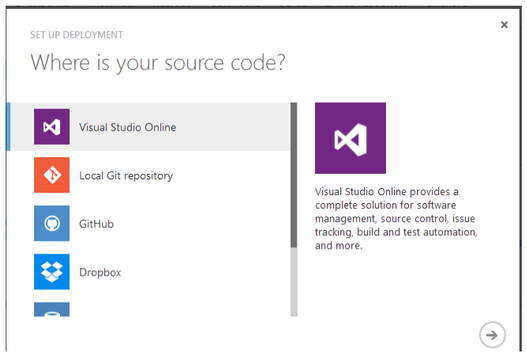


Figure 6: Select Source Control repository VSTS (VSO)

As a security measure, it asks us to accept the request to connect our application and VSTS account so that they can be accessed by Azure service on our behalf.

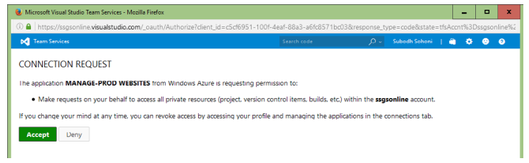


Figure 7: Authorize Azure to link to your VSTS Account

After the authorization has been provided by us, it links our application to the selected team project from our VSTS account. It also creates a build definition (XAML Build, not yet the new Scripted Build) that has Continuous Integration trigger set.

It also deploys the changes to the Service App as soon as the build is successfully completed.

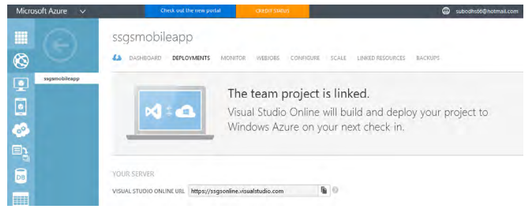


Figure 8: Service App linked to VSTS for automated build and deployment

Continuous Integration

We can make a small change now and check-in that. We can view the status of deployment while the build is executing.

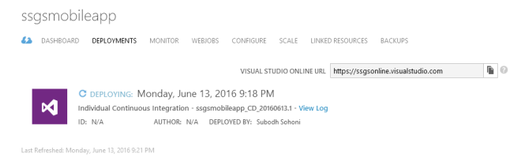


Figure 9: Continuous Integration – Deployment status continuously updated

We will also come to know when the build is deployed and which build is the active build.

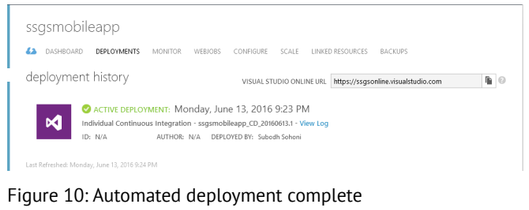


Figure 10: Automated deployment complete

After the new build is deployed, if we detect any major bug in the application, it is possible to revert back to earlier build, in one click.

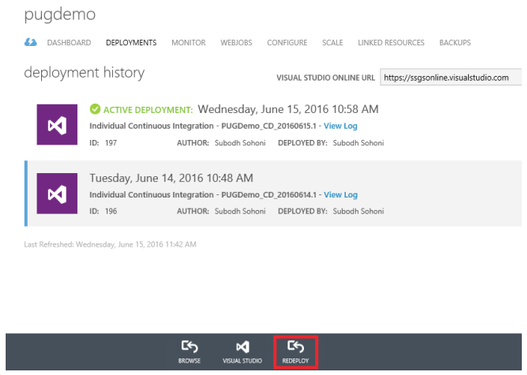


Figure 11: Deployment reversal

Create and Download UI Application

Now that our backend service project is ready, let us create a UI application that will connect to this service application. To do so, we will go back to the new Azure portal. We will open the blade of the application and go to the Quick Start blade of that. Here we can see various client formats for iOS, Android and Windows platform with or without Xamarin. We will select Xamarin.Forms format so that we can create common (shared) code for multiple platforms. We can now see the download button for the new client application. By clicking on it, we can download the client app in ZIP form.

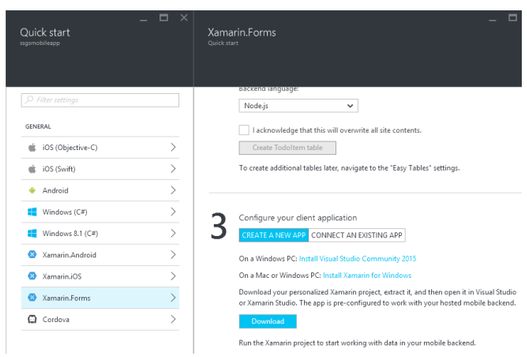


Figure 12: Create and download client app

Once the application is downloaded as a ZIP file, we can unzip it and open the solution of the client application. It is already wired to the back end Service App that we had created earlier.

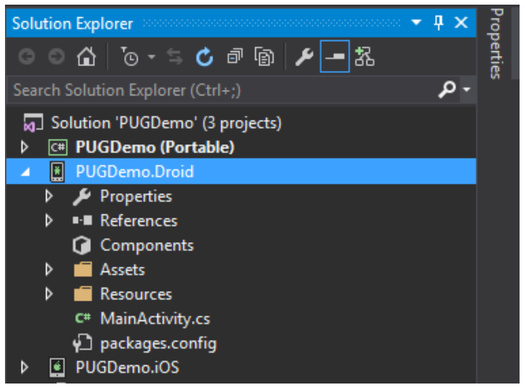


Figure 13: Client application opened in Visual Studio 2015

We can now put it in the same source control so that entire team can start development of that.

Summary

In this article, we took an overview of how to create an entire mobile application that uses Azure App Service to host its back end logic. We used Xamarin to create the front end of the application. We also used VSTS to do source control, build and automated development of the service app to Azure App Service ■

About the Author

subodh sohoni

Microsoft® Most Valuable Professional

Subodh *is a consultant and corporate trainer. He has overall 28+ years of experience. His specialization is Application Lifecycle Management and Team Foundation Server. He is Microsoft MVP – VS ALM, MCSD – ALM and MCT. He has conducted more than 300 corporate trainings and consulting assignments. He is also a Professional SCRUM Master. He guides teams to become Agile and implement SCRUM. Subodh is authorized by Microsoft to do ALM Assessments on behalf of Microsoft. Follow him on twitter @ subodhsohoni*

Ravi Kiran

TRANSPILING **ES6 MODULES:** using Babel

ES6

ECMAScript 2015 (also known as ES 2015 or ES6) is the current version of JavaScript, and its biggest update till date. It comes with a number of language improvements and API updates to make JavaScript developers more productive. The specification of the language was frozen in June 2015. Browsers have started implementing the new features, although they still need some time to implement all the features. Meanwhile, as we saw in previous articles on ES6, transpilers have filled in the gaps and have allowed us to use these new features, till the browsers implement them all.

There are a number of transpilers that convert the ES6 code to browser readable ES5 code. The most popular one these days is **Babel**. It was named 6to5 earlier, but then got renamed to Babel to make the name generic. The other popular alternatives are Traceur from Google and TypeScript, the typed JavaScript language from Microsoft.

In this article, we will use Babel to transpile ES6 code to ES5.

The reason for choosing Babel is, amongst all other transpilers, Babel supports the most number of ES6 and ES7 features. The ES5 code it generates after transpiling the ES6 code, can easily be read and understood by any average JavaScript developer, as it doesn't add any special variables or hacks in the converted code. In this article, we will see how to setup an environment and transpile the ES6 code into ES5 using Babel.

About the sample ES6 application

The sample used for this article is a simple application that uses GitHub's REST API to fetch the list repos under a GitHub account and displays them in an HTML table. The application consists of three JavaScript files written in ES6 and they perform a different set of tasks for the application.

The following figure shows the folder structure of the application:

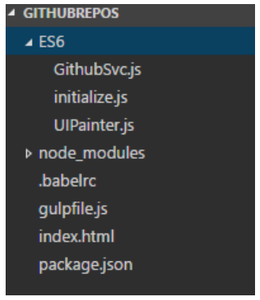


Figure 1: ES6 application folder structure

The file *index.html* is the starting point of this application. It has to load the JavaScript files to display the GitHub repos.

The following is the list of JavaScript files of the project and a brief description of the task they perform:

- **gitHubSvc.js**: Queries the GitHub REST API using jQuery and gets the list of repos under an organization
- **uiPainter.js**: Accepts the data that has to be presented and displays it in the target HTML element
- **initialize.js**: This files uses both of the above files and wires them up to the page

All of these files use the import/export syntax of the ES6 module system and they use ES6 features to achieve their tasks. You can check the code in ES6 folder of the sample on GitHub.

Transpiling using Babel

Babel can be used to transpile the JavaScript code through its command line interface (CLI) or can be used as part of the JavaScript build tools like npm scripts, Gulp and Grunt. We will see how to transpile using Babel through command line and using Gulp.

Using Babel CLI

To use Babel through command line from any folder in the system, we need to install the global npm package of Babel. The global package has to be used only to play around with the features of the tool. To use it in a project that is being built by a team of people, we should install it in the project. We will see both of these use cases.

Using Global Babel Package

To install Babel globally, we need to use the following command:

```
> npm install -g babel-cli
```

To check if Babel is installed properly, you can run the following command to check the version of Babel package installed on your system:

```
> babel -V
```

If this command is unsuccessful, you need to try installing Babel again. Once Babel is installed, we can use the command with different options to transpile the JavaScript code.

To transpile a single file, you can pass name of the file to the Babel command:

```
> babel script.js
```

This command transpiles the code in script.js file and displays result in the console. If you want to save the output of this command into a file, you need to use the --out-file option of the Babel command and provide a path of the target file.

```
> babel script.js --out-file script-transpiled.js
```

This command doesn't convert the ES6 code to ES5. To do so, we need to install the ES2015 preset of Babel and make it available to Babel through its configuration file, .babelrc. Babel's ES2015 preset contains all of the plugins required to convert every new API and feature of ES6 to its ES5 equivalent.

You can see the list of plugins installed with the ES2015 preset on this page. The preset is an npm package, we can install it using the following command:

```
> npm install babel-preset-es2015 –save-dev
```

We need to store the configuration required for Babel in a file named .babelrc. The following JSON snippet shows how the preset has to be loaded:

```
{
  presets: ["es2015"]
}
```

If the application needs any other preset, we can mention it in the above array and Babel will automatically use them. Now the command will produce an ES5 equivalent of the ES6 code present in the script.js file.

Using Local Babel Package

If you want to use Babel through a local package and share it with your team, you need to install it as a local package and save it in the *package.json* file. We need both Babel and babel-cli packages to use the utility as a local package. The following command does this:

```
> npm install babel babel-cli --save-dev
```

The local package installs the commands to the *node_modules/.bin* folder and these commands can be accessed using the scripts section of the *package.json* file. We can register the command to transpile the JavaScript file under the scripts section as follows:

```
scripts: {
  “transpile”: “babel script.js --out-file script-transpiled.js”
}
```

We can invoke this script using the following command:

```
> npm run transpile
```

Now let's start working on transpiling the sample code. If you want to follow along, download the folder *GitHubRepos_before* from the source code of this article and install the node modules. Once the installation is complete, follow the instructions discussed in the next section.

Transpiling ES6 Modules to AMD using CLI

Let's transpile code of the sample application to ES5. We will discuss how to do it using the command line interface and then we will do the same thing using Gulp.

If you use the ES2015 preset and follow the process mentioned in the previous section to transpile the ES6 code, you will notice that the ES6 modules used in the files are converted into CommonJS style modules. CommonJS is a JavaScript module system designed by the open source developers at the CommonJS group. It provides the modules with the objects *require* and *exports* to import and export the modules respectively. It loads the modules synchronously and the objects exported from the other modules can be used in the consuming module. Visit the CommonJS wiki page to learn more about the module system.

The ES2015 preset includes the Babel plugin to convert the code into CommonJS style module system, so this plugin detects the presence of modules in the ES6 file and converts it into CommonJS module.

Alternatively, we can convert the ES6 modules into AMD, SystemJS or UMD modules by installing the corresponding Babel plugin and including it in the *.babelrc* file. The AMD (Asynchronous Module Definition) style module system defines a way to load the modules asynchronously. A module can use a set of other modules as its dependencies, and these modules are loaded asynchronously without waiting for the previous modules to finish. The module using these dependencies is executed once all of these modules are ready.

Let's convert the ES6 modules into AMD style modules. The following command installs the npm package containing the plugin for this purpose:

```
> npm install babel-plugin-transform-es2015-modules-amd
```

We need to include it in the plugins section of the .babelrc file. Add a new file to the project and change its name to .babelrc. Paste the following snippet into it:

```
{
  "presets": ["es2015"],
  "plugins": ["transform-es2015-modules-amd"]
}
```

Now, running the following command will transpile the ES6 files into AMD style modules and store them in the dist/temp folder.

```
> babel ES6 –out-dir dist/temp
```

To load this code in the browser, we need a library that would help us in loading the AMD modules on the browser. Require.js is one of the libraries used to load AMD modules in the browser. It has to be installed using npm. The following command installs this library:

```
> npm install requirejs –save
```

Let's transpile the modules in the application to AMD now. The following command does this for us:

```
> babel ES6 --out-dir dist
```

This command reads all the files under the ES6 folder, converts them to ES5 with AMD syntax and copies the resultant files into a folder named dist. In order to use this command repeatedly, we can store it in the scripts section of package.json.

```
"scripts": {
  "transpile-amd": " babel ES6 --out-dir dist"
}
```

This command can be invoked from the command line as follows:

```
> npm run transpile-amd
```

To load the converted files on browser and see them in action, we need to include the require.js library and ask it to load the first file of the application, which is initialize.js. For this, we need to add the following statement to the index.html file:

```
<script src="node_modules/requirejs/require.js" data-main="dist/initialize"></script>
```

The *data-main* attribute in the above snippet is used to load the first AMD module. Every file is treated as a module, so this snippet loads the file *initialize.js*. As this file needs the other two files, require.js loads them asynchronously before running the code inside the module of the file *initialize.js*.

Run the application and you will see the page loading the list of projects by AngularJS team (I hard coded the organization as angular in the code).

Name	Full Name	Owner	Language	Forks/Watches	Issues	Visit
samples	angular/samples	angular	Ruby	9/1	0	Visit
angular.js	angular/angular.js	angular	JavaScript	23813/49469	1116	Visit
calculator-sample	angular/calculator-sample	angular	JavaScript	6/17	0	Visit
angular-seed	angular/angular-seed	angular	JavaScript	6510/11360	81	Visit
angular-jquery-ui	angular/angular-jquery-ui	angular	JavaScript	26/93	2	Visit
angularjs.org	angular/angularjs.org	angular	JavaScript	204/254	15	Visit
angular-phonecat	angular/angular-phonecat	angular	JavaScript	4245/2397	17	Visit
jstd-jasmine	angular/jstd-jasmine	angular	JavaScript	1/2	0	Visit
peepcode-tunes	angular/peepcode-tunes	angular	JavaScript	54/212	2	Visit

Transpiling ES6 code using Gulp and Babel

Gulp and Grunt are widely used as task runners in front-end applications. They make it easy to configure and run tasks to perform many of the tasks that we otherwise had to perform manually. Out of the two, I personally like Gulp for its API and the asynchronous approach.

Babel has packages for both Grunt and Gulp that can be used to achieve everything that is achievable through the commands that we saw in the previous section.

Like Babel, if you want to use Gulp from anywhere on the system, then we need to install it globally using the following command:

```
> npm install -g gulp
```

Otherwise, we can use it from the local Gulp package as we did in case of Babel.

First things first, let's install the required npm packages to perform Babel's tasks with Gulp and to use require.js with Gulp. Following are the packages we need:

- gulp: A local copy of gulp
- gulp-babel: This package enables us to use Babel in a Gulp task

We can install both of these packages using a single command, the command is shown below:

```
> npm install gulp gulp-babel --save-dev
```

Add a new file to the root folder of the project and name it *gulpfile.js*. Let's get references of the npm packages we need in this file.

```
var gulp = require('gulp'),
  babel = require('gulp-babel');
```

Before writing a task to transpile the code in the sample application, let's understand how Babel works with Gulp. To transpile a file using Babel in Gulp, we need to load the file and pipe the Babel task into it. The following snippet demonstrates how an imaginary ES6 file named *sample.js* can be transpiled in a Gulp task:

```
gulp.task('transpile', function(){
  return gulp.src(['sample.js'])
.pipe(babel())
.pipe(gulp.dest('trasnpiled-sample.
js'));
});
```

The above task transpiles the code in the file supplied and stores the resultant file in a file named *transpiled-sample.js*. As we invoked the Babel package without passing any values into it, it will use the configuration stored in the file *.babelrc* in the current folder. If it doesn't find the file, the transpilation will happen according to babel's default configuration. The above task can be invoked from the command line as follows:

```
> gulp transpile
```

We can pass Babel's configuration to the task, instead of storing it in the *.babelrc* file. Remember not to mix them both, as at times it may result in some odd results, that are hard to spot and fix. The following snippet passes the configuration into the Babel task:

```
return gulp.src(['sample.js'])
.pipe(babel({
  "presets": ["es2015"],
  "plugins": ["transform-es2015-modules-
  amd"]
}))
.pipe(gulp.dest('trasnpiled-sample.
js'));
```

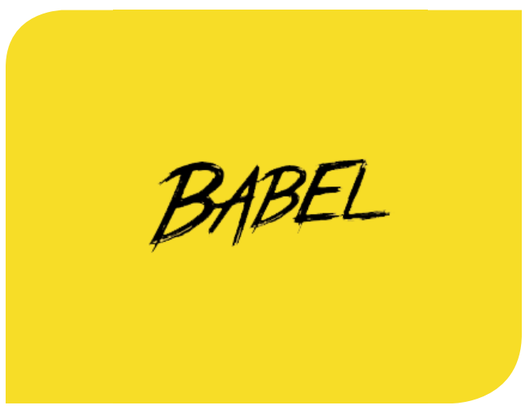
Now that we understood how to use Babel with Gulp, let's write a Gulp task to transpile the files in the sample application. The task that we are going to write is similar to the sample task shown above; we just need to change the paths in it. The following snippet shows the task:

```
gulp.task('transpile', function () {
  return gulp.src(['ES6/*.js'])
    .pipe(babel())
      .pipe(gulp.dest('dist'));
});
```

We can run this task using the following command:

```
> gulp transpile
```

This command converts the ES6 files into ES5 files with AMD and copies the files into the dist folder. As we added reference of require.js library with the initial script to the HTML file in the previous section, we don't need to make any changes. Run the application and you will see the same result as we saw in the last section.



Conclusion

The ES6 specification added a much-needed feature of Modules to the JavaScript language [http://www.dotnetcurry.com/javascript/1118/modules-in-ecmascript6]. Though they are not entirely supported by the browsers yet, transpilers like Babel help us in using them today. As we saw, we can use Babel directly or with a task manager like Gulp to convert the ES6 modules into AMD style modules. We can convert it to other types of modules as well ■

Download the entire source code from GitHub at bit.ly/dncm24-transpile-es6

About the Author

ravi kiran

Microsoft® MVP Most Valuable Professional

Rabi Kiran (a.k.a. Ravi Kiran) is a developer working on Microsoft Technologies at Hyderabad. These days, he is spending his time on JavaScript frameworks like AngularJS, latest updates to JavaScript in ES6 and ES7, Web Components, Node.js and also on several Microsoft technologies including ASP.NET 5, SignalR and C#. He is an active blogger, an author at SitePoint and at DotNetCurry. He is rewarded with Microsoft MVP (ASP.NET/IIS) and DZone MVB awards for his contribution to the community.

Kunal Chandratre

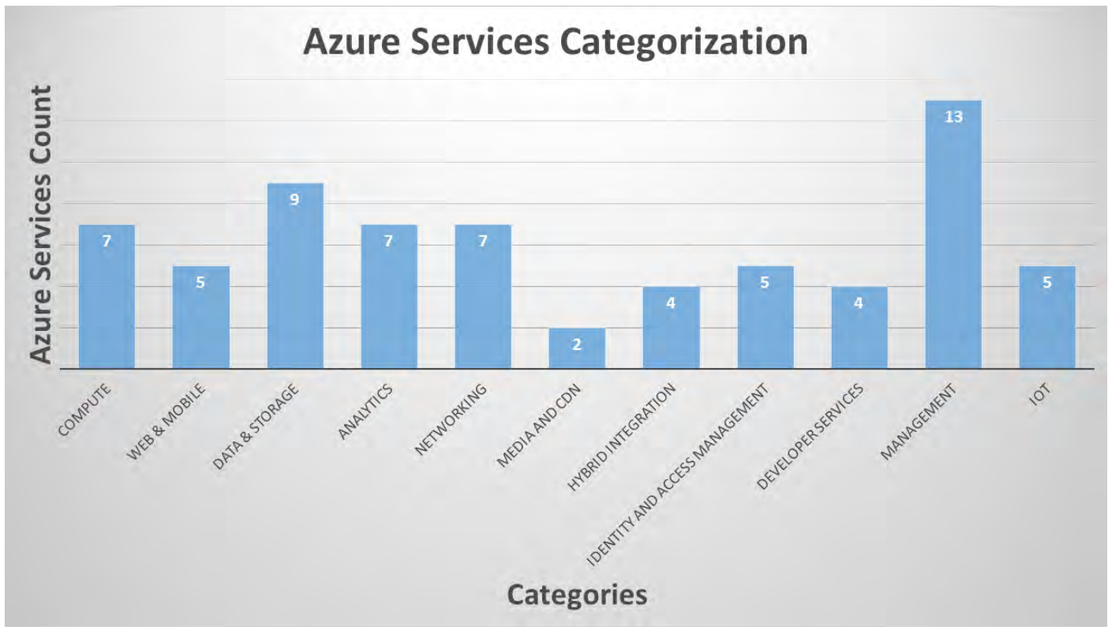
Demystifying the Azure platform

The Microsoft Azure platform is a cloud computing platform which provides a wide variety of services to build scalable applications. Over the years, the Azure platform has matured and is evolving at a rapid pace compared to any other cloud platform in the market. Numerous services and features are being added to the Azure platform on a constant basis. This also creates a challenge. How does one keep track of the capabilities that Azure services are offering?

This article is an attempt to provide some quick access information, important links and other necessary details about all the services that are offered today (as of this writing) on Azure.

Consider this article as a one stop solution to get crisp and to the point information about all services present on Microsoft Azure as of today.

Let's get started



Half Century is already crossed!!

Yes, you read it correct. As of today Azure offers 50+ services. By the time you read this article, there could have been a couple of more services released in private preview or even in General Availability.

We can divide these services into eleven categories at a high level as seen in the Chart on the previous page.

In this article, we will try and address the following aspects of each service per category –

1. What is the intent of Service

2. Amazon Web Service (AWS) equivalent service name. [Yes, most of the times I get queries regarding equivalent services between Azure and Amazon so thought of adding it here, wherever applicable.]

Two deployment models in Azure

Before going any further I would like to first highlight an important aspect of Azure deployments. As of today, services on the Azure can be deployed either by using –

1. Azure Service Management (ASM) or Classic model

2. Azure Resource Manager (ARM) model

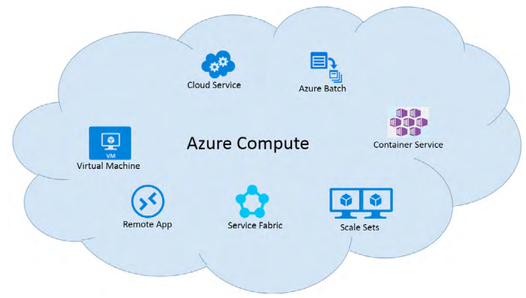
Classic model is the older way of doing deployments, whereas ARM is the future. Classic focuses more on the "How" part of Azure provisioning and deployments, whereas ARM focuses on the "What" part of Azure provisioning and deployments.

In Classic model, you were managing each Azure resource individually, whereas in ARM, you manage all the resource grouped under "Resource Groups" entity which make the management tasks simpler and easy.

Going forward, in this article, we will not separate the service information as Classic and ARM, *but only focus on the intent of the service*. The deployment models changes are mostly with respect to management and provisioning semantics.

Compute Services

The Compute services category has in total 7 Azure services as shown here –



Virtual Machines

Virtual Machines (VMs) is an IaaS offering. A VM is used to provision Windows and Linux virtual machines as per the requirements. The service has pre-built images created by Microsoft, partners and community. As you provision VMs, you have complete control over the OS. Although that's an advantage, it also means it is your responsibility to do management of VMs, such as OS upgrades.

Azure Virtual Machines can be provisioned from Gallery images or custom developed images. The underlying data disks and OS disks are always preserved in Azure Page blob hence the VMs are persistent. Using Powershell, Azure portal, CLI or REST API; users can provision the VMs either in classic or ARM model.

Amazon Web Service (AWS) equivalent service name – EC2

Cloud Service

Azure cloud service is a PaaS offering from Azure and specifically designed for hosting web applications, background processing applications [similar to traditional windows service applications] and Azure IaaS workloads i.e. Virtual Machines. These services can be deployed using only the Classic model. Cloud services don't support ARM deployments.

Cloud Service in Azure is a container under which applications run. The web application in cloud service is termed as "Web Role" whereas background processing applications are termed as "Worker Role".

Key differentiator between cloud service roles and IaaS VM provisioning is that the VMs that you get as a part of role instances, are 'not persistent'. That means if you make any role instance changes on VM like creating text file on C drive, it will go away in case roles get recycled. So state and data will not be permanent. Whereas, VMs of IaaS workload provisioned in cloud services are 'persistent'. That means any changes to data and state will be permanent with exception of D drive, which is a temporary drive.

Amazon Web Service (AWS) equivalent service name – Elastic Beanstalk.

VM Scale Sets

Traditionally horizontal scaling in Azure VMs is supported but the major hurdle is "pre-configuration of VMs". Unless you already have some VMs pre-configured, auto scale is not possible. In addition to this, the scale number is limited to pre-configured VM count. To overcome this problem and free yourself from pre-configuration of VMs for scalability, VM Scale Sets can be used.

Amazon Web Service (AWS) equivalent service name – Auto Scaling

Remote Apps

Remote apps is an Azure service which can be used to provide remote access to applications independent of devices and platforms. The foundation behind the Remote app is Microsoft Remote Desktop Services with cloud enablement. The remote apps today have the Office application pre-configured, plus you can also run custom desktop applications on them. VNET connectivity and domain join support makes it a powerful choice for running enterprise applications with full control over application topology. Basically you can deploy a windows app today in remote apps, and can access it on any type of device. If you are interested in Azure Remote apps, read my article Run Desktop applications in the Cloud using Azure Remote Apps [bit.ly/dncm25-azure-remote-apps]

AWS equivalent service name - AppStream

Azure Batch

Azure Batch is a High Performance Computing solution (HPC) on Azure. It is also known as "Big Compute". As the name indicates, this service is for scenarios where you need huge computing power, but not necessarily scenarios where you need big I/O performance and disk capacity [In such case you may think about Big Data – Hadoop]. When the task is "Intrinsically or embarrassingly parallel" it should be considered as a good candidate for Azure Batch.

Intrinsically parallel means a perfectly parallel problem where no effort is needed to divide the problem into a number of tasks for processing in parallel. These are the problems where no dependency is present on other tasks and no communication is required between the tasks running in parallel. For example - Media encoding and transcoding, Image rendering and processing.

AWS equivalent service name - No equivalent service in AWS exists as of this writing.

Service Fabric

Service Fabric is based on "Microservices Architecture". As of today, we build enterprise level web applications based on a 3-tier architecture where in the UI, Business Layer/ Data Access Layer, and Database are separate; but in true nature of implementation, they are still tightly coupled and monolithic. In contrast to traditionally divide and build an application in 3-tiers, a microservice application is divided in independent components which are called as "Microservices". The emphasis is on the fact that applications should be comprised of small enough services which truly reflects its single purpose, such that every micro service implements a single function. Every service has REST contracts defined so that other services can communicate and share the data. In a way, you are achieving a great level of loose coupling in the application.

So the Microservices architecture implementation is nothing but "Service Fabric". This has inbuilt support for massive scale, hybrid deployment model, 24x7 availability and most importantly, support for DevOps to manage health based upgrades and automatic rollbacks.

AWS equivalent service name - No service exists today.

Container Service

Basically a container is an isolated place where application runs without affecting rest of the system, and vice-versa. Containers are future of virtualization techniques. The current virtualization techniques are of hardware virtualization, whereas container is OS virtualization.

Azure container service can be used to create a cluster of virtual machines that are pre-configured to run a containerized application. Today ACS allows you to create cluster of "Docker Swarm" and "Apache Mesos". A major use of Azure container service I see in the future is in implementation of MicroServices based offerings.

AWS equivalent service name - EC2 Container Service, EC2 Container Registry

Web and Mobile

Probably this is the most confusing category because Azure Service names in this category are similar to each other. This category has a total of 5 services as shown below -

The confusion usually happens with the following service names –

1. Mobile Apps, Mobile Service and Mobile

Engagement

2. API Apps, App Service and API Management

I hope there will be no confusion once you understand the intent below.

App Service

Earlier, Azure Web Sites and Mobile Services were provided as a separate service on Azure. To cater to needs of "Cloud first, mobile first" strategy, Microsoft renamed Web sites and mobile services to Web Apps and Mobile Apps respectively and brought them under one umbrella service called as "App Service".

On top of this, Microsoft also added two new services under "App Service" named as "Logic/ Workflow Apps" and "API Apps". Now in the near future, "Azure Functions" will also get added as a part of the App Service. So in essence, App Service can be viewed as follow –

Web Apps + Mobile Apps + Logic Apps+ API Apps + Functions = App Service

App Service provides powerful capabilities such as built in DevOps, Continuous integration with Visual Studio Team Services/ Github, staging and production slots, automated patching and so on.

AWS equivalent service name - Auto Scaling

Web Apps

Web Apps service is a PaaS offering on Azure and specifically used for hosting web applications. The difference between various offerings used today for web application hosting is as follows –

Virtual Machines	Customized Persistent virtual machines with FULL control over OS
Cloud Service Roles	Non persistent VMs with control over OS up to SOME EXTENT
Web Apps	Helps in building web application for various platforms with NO control over underlying VM and OS.

So Web Apps is a hosting provider solution on Azure similar to other hosting providers where you can create apps from the Gallery for various platforms like Drupal, Wordpress, Joomla, ASP.Net MVC etc.

Unlike *Worker Roles* of cloud services where the application runs on dedicated VMs, the background jobs in Web Apps are called *Webjobs* that run in the same context of Web App, and you are not charged anything extra for it. Web Apps should be your first choice when you are planning to host a web application. If any of the application scenarios are not getting fulfilled with Web Apps capabilities, then you can explore options of Cloud Service roles and virtual machines.

AWS equivalent service name - Elastic BeansTalk

API Apps

API Apps is about hosting your REST APIs. Using this service you can easily create, consume and call APIs that are developed in any framework like ASP.NET Web API or similar. You can also bring your existing APIs as-is irrespective of the language in which it has been created. The key features offered are Authentication, CORS support and API metadata.

You can make the API app hosted REST APIs available for your internal consumption or you can publish them on Azure Marketplace and make some revenue out of it. For example, let's say you create a REST API for getting address information of a person based on his name within a city, then it can be used in any type of application and you can charge on a per call basis. Just a thought!

AWS equivalent service name - No service exists today.

Mobile Apps

Mobile Apps on Azure App Service is nothing but a MBaaS (Mobile Backend as a Service) offering. Earlier, building Mobile Apps was never easy as there were so many infrastructure configurations required like database, notification services, setting up single sign-on, other authentication services and so on.

Mobile Apps makes all of these services available on your fingertips. It offers various services such as

- Authentication and Authorization [using Azure AD, social providers etc.]

- Data access [mobile friendly OData v3 data sources linked to Sql Azure, SQL on VM, NoSQL providers such as Table Storage, Mongo DB, Document DB]

- Offline sync between data on device and backed data store

- Push notifications capabilities using Azure Notifications Hub

- Development SDKs for native development on iOS/ Android/ Windows, Cross platform development using Xamarin, hybrid application development using Apache Cordova and so on.

AWS equivalent service name - Mobile Hub

Mobile Services

This service is superseded by Mobile Apps service and its intent has already been described above.

Logic Apps

Logic apps can help a technical user or developer to build an automated process execution and workflow, using visual designer. The best part is you don't have to write a single line of code for scenarios which are pre-built into logic apps.

Logic app has "Managed API" connectors like Yammer, SMTP, Sharepoint Online, Outlook, OneDrive, Office 365, CRM Online and so on which can be used in building any type of workflow application in Azure portal itself.

There are "Hybrid connectors" also available to have connectivity with SAP, Oracle, DB2 and so on. The connectors are nothing but Azure API Apps whose focus is only on connectivity. So essentially you can create your own API App and have it used in creating logical workflows and business process automation using Azure Logic apps. In that case, it can be called as "Custom connector".

AWS equivalent service name - Lambda

Azure Functions

As the name indicates, it is a solution for running small pieces of code having single responsibility and you are charged per execution. Azure function can contain code that will get triggered by an event.

In my opinion, in future, Functions will form various components of Microservices [an event driven independent service architecture] built on Azure.

These functions can also be helpful in filling the gaps of Logic app implementation. Most of the times, developers were running into cases where Logic App Workflow Definition Language was reaching its limitations. In such cases, writing custom API App and adding it to Logic app, was the only option. Azure functions can fit into these gaps.

AWS equivalent service name - Lambda

API Management

A common architecture pattern for all modern apps today is to expose business logic through APIs and expose endpoints over HTTP using REST protocol. This approach enables the consumption of APIs from any language using any platform. Although building a REST API building is fine, but to make them scalable, secure and policy enabled etc., is difficult and this is where API Management comes into the picture. There are 3 main components –

1. API Gateway:
Responsible for accepting API calls and route them to backend services, Verify API keys and JWT tokens, enforce usage quota and rate limits, caching, logging, monitoring.

2. Publisher portal
This is an administrative interface where we can setup API program. It can be used to define API schema, package API as a product, setup usage and access policies, analytics.

3. Developer portal
Provides access to developers for management where they can read API documentation, try API sample from the console itself, get API keys, access analytics for their own purpose.

AWS equivalent service – API Gateway

Mobile Engagement

This service offers various features so as to maximize user retention and monetization. It is a SaaS offering on Azure which can be used to get data-driven insights in application usage, real-time user segmentation, in-app messaging and notifications. This service is specifically for the digital marketers, however at the same time it can be used by mobile app owners as well who want to increase the usage, user retention, and monetization of mobile apps.

Few scenarios where Mobile engagement can be used are as follows –

1. Welcome campaign – This can be used to encourage account creation for users by providing welcome invitation and multi-language capabilities.

2. Rating boosts – Send a message to users and encourage them to provide ratings. This directly increases the popularity and discovery of the app in market.

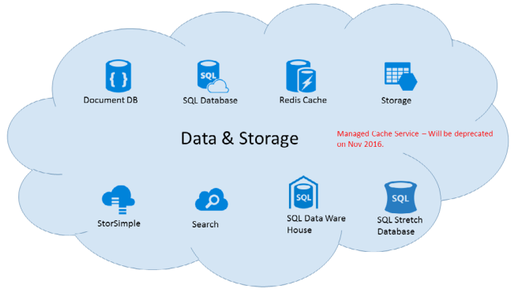
3. Sorry – Send automated messages to users on app crash and errors that affect them most. This influences user retentions directly and users can be patient till you fix the problem.

Similarly, there can be many such scenarios where Mobile engagement can be a useful feature.

AWS equivalent service – Mobile Analytics

Data and Storage

This category has 9 services as shown below –



Document DB

DocumentDB is a NoSQL document database-as-a-service offering which stores all the information in JSON format. Those who have worked with Mongo DB can relate DocumentDB as an offering from Microsoft Azure in the NoSQL space. DocumentDB enables complex queries using SQL syntax based language, supports transactions across documents with a familiar programming model of writing stored procedure, user defined functions, triggers written in JavaScript etc. Along with JavaScript support, it offers all the operations with REST APIs.

AWS equivalent Service – DynamoDB

SQL Azure Database

This is a PaaS offering for SQL database. Some people also term it as "Database as a Service".

Being a PaaS offering, you don't have access to the underlying server but you can communicate via endpoints. This is available in 3 tiers – Basic, Standard and Premium.

Depending on the chosen tier, you get Performance units. This is not similar to SQL server running on VM hence you can't configure SSRS, SSIS, SSAS, SQL agent etc. on it. Moreover cross-database queries and distributed transaction is also not supported.

At the same time the High Availability, backup and Disaster Recovery strategies of SQL Azure database is awesome and completely automated. It reduces a lot of headaches on the maintenance part. Today various tools and options are available by which you can migrate from on-premises SQL server to SQL Azure DB offering or vice versa which makes this offering very powerful and suitable for production workloads.

AWS equivalent service – RDS.

Redis Cache

Azure Redis Cache is based on the popular open source Redis Cache. Most of the cache offerings usually deal with only key-value pair for storing and retrieving information from cache. On the other hand, Redis supports data types on which you can perform atomic operations like append, increment, union and difference, sorting etc. which may not be possible with traditional cache offerings. The most popular use of Redis cache is seen in storing session information in ASP.NET web applications.

AWS equivalent service – ElastiCache.

Managed Cache Service

This service will be deprecated in Nov 2016 and the alternative is to use REDIS Cache offering on Azure.

Storage

Azure storage offers various storage services for your business needs. It is massively scalable data store and can store data consisting of hundreds of Terabytes. It provides 4 types of storage options –

1. Blob Storage –

This stores unstructured data. It can be text or binary data. This is also referred as Object store and has two types –

a. Page Blob – Used for storing backed Vhd files of Azure VMs. Targeted for Random read/ write operations.

b. Block blob – Used for storing any type of file or binary data. Targeted for sequential read/ write operations.

2. Table Storage

Stores structural dataset and is a NoSQL key-value pair based store.

3. Queue Storage

Provides reliable messaging solution which can be used in various integration scenarios.

4. File Storage

A File Storage option for applications using standard SMB protocol. This is specifically used for making file storage available as local drive in virtual machines. Blob storage is accessed over HTTP and can't be mounted/ accessed as local drive; this is where Azure files help.

AWS equivalent Service -

AZURE SERVICE	AWS SERVICE
STORAGE	Glacier
BLOB	S3
QUEUE	Simple Queue Service
TABLE	DynamoDB and SimpleDB
FILE	Elastic File System

StorSimple

StorSimple is a cloud integrated storage product that allows users to store heavily used data on premises or locally, and put older and less active data on Azure. SotrSimple is a Virtual Storage Array solution. Basically it is a software version of storage that can run independent of dedicated hardware, in a VM or cloud services.

StorSimple uses the concept of "Storage Tiering". The data is arranged in logical tiers depending on the age, usage and relationship to other data. Data that is the most active (termed as hot data) is stored locally using SSD(Solid State Drives), the data used occasionally (termed as warm Data) is stored on HDD (Hard disk drives) locally or in data center on premises and less active data (cold data) is moved to Azure.

AWS equivalent Service – Storage Gateway.

Search

Search is the primary requirement of any application because it's simple, powerful and requires no training to use. However providing search option can be challenging for developers and to build your own search engine involves too much effort. Even if we think about 3rd party search tools they are expensive and configuration/ installation required is too much. So typically a search service is needed where developers don't have to manage anything but can be used directly in applications running on premises or in cloud. This is what Azure Search provides.

It can be used for search operation on SQL Azure Database, SQL server on VM and DocumentDB or NoSQL store running on Azure VM. It is important to note here that Azure search is entirely focused for developers. It can't be used directly by end users from the UI. Everything today about Azure search needs to be implemented using REST API or SDK and application developers are free to create UI of their choice and need.

AWS equivalent service – ElasticSearch Service.

SQL Data Ware House

This is a "Data warehouse as a service" offering from Azure. This service is built on Massively Parallel Processing (MPP) architecture. The purpose of the service is similar to the purpose of any data ware house system but this adds cloud enablement. So you can push data from various different sources like SQL/ NoSQL and can run the analysis, queries on it to get meaningful data and outcome out of it. Most importantly, the query language is Transact SQL which most developers are already familiar with.

The "Azure SQL Data Warehouse" can be used with other tools like PowerBI for visualizing data, Azure Data factory for event processing, Machine Learning for predictive analysis, HDInsight for analytics service, and for similar tools in the future.

AWS equivalent service – Redshift.

SQL Stretch Database

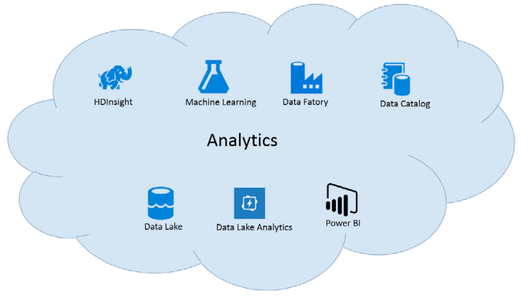
SQL Server 2016 added a new feature in its release which enables you to store a portion of a database in Azure, which is known as Stretch database. You can move Warm and Cold data from on-premises to Azure and the moved data remains online for a query from the application, which may not be the case with traditional cold data storage options.

To identify which database and tables are good candidates for stretch database implementation, you can use "Stretch Database Advisor" tool of SQL server 2016.

AWS equivalent service – None.

Analytics

Following are the services which are part of Analytics category on Azure –



HDInsight

HDInsight is the Azure cloud implementation of the Apache Hadoop technology stack which is a popular solution for Big Data analysis.

Big Data refers to any large body of digital information which can contain data collected from various sources. Some examples include Aero plane Black Box data, Social Media data, Stock market data, Search engine data, Power grid data, e-commerce data, sensor data from industrial equipment and so on. Big data can be historical (stored data) or real time (streamed directly from the source).

To use this data and to come up with usable, actionable presentation, we need analysis tools capable of handling such massive amounts of data. Apache Hadoop is an open source popular framework which offers various tools to perform such Big data analysis.

Hadoop being open source, many companies have created their own stack of Hadoop offering called as Hadoop distribution. Out of these, a popular distribution is HortonWorks.

HDInsight uses HortonWorks Hadoop distribution and offers entire Hadoop eco-system components as cloud enabled on Azure platform.

AWS equivalent service – Elastic MapReduce (EMR).

Machine Learning

Data has secrets when it is presented in a big volume. This big volume data can be examined to find patterns. Ultimately these patterns can help you to solve a problem, predict future events in advance and get prepared for it. However examining such a big volume of data to identify patterns is a complex job and this is what is made easy by Azure Machine Learning.

So typically machine learning uses computers to learn from existing data in order to forecast future behaviors, outcomes and trends. In short, it provides "Predictive Analysis" to make applications and devices smarter.

Azure ML not only provides tools to model predictive analytics, but also provides a fully managed service that you can use to deploy your predictive model as web service and monetize it.

AWS equivalent service – Machine Learning

Data Factory

Analytical data scenarios always have many moving parts and various inflow points. For example, building data inside data warehouse needs interaction with various systems and a variety of locations like cloud and on-premises, to get the data. It is possible to establish this process manually, but if the same process needs to be done on a regular basis, then it is a good candidate for automation. Azure Data Factory lets you automate this data collection process based on schedule and from various sources like SQL Server PaaS, SQL Server on premises, Azure blob, tables, HDInsight Hadoop systems and so on.

So essentially, Azure Data Factory is for movement of data between sources termed as "Data movement activities" and on top of that, you can run Analytics termed as "data transformation activities" using tools such as HDInsight hive, Pig, streaming or Azure Machine Learning. So data factory is not for storage of data, but for movement, and if required, transformation.

AWS equivalent service – Data Pipeline.

Data Catalog

Data Catalog is basically a data discovery service. The purpose of the data catalog service is to provide a way to identify the data sources and then search them in an organized way.

As a part of the registration process, users registers the data sources from which metadata structures gets extracted automatically, whereas actual data remains in the data source. Once the metadata is in catalog store, users can take actions like search, filter, annotate, and add additional information about data sources and metadata, and so on. Additional information added to data sources helps those who have just joined the system/ organization to identify the intent of the data source.

AWS equivalent service – None.

Data Lake Store

Data Lake Store is a repository on Azure that holds all the data in its raw format. This way Data scientist, Data analysts can store data of any shape, size and can perform all types of processing and analytics across platforms and languages. On top of it, the data stored in Data lake store can run Hadoop, HDInsight or Data Lake analytics to find patterns of data.

The difference between Data Lake and SQL Data Warehouse –

Azure SQL Data Warehouse enables you to store the data in a structured manner and is coupled to one or more schemas.

In Azure Data Lake, the incoming data is stored as-is in raw format, and analysis is done on top it with various tools.

AWS equivalent service – Kinesis Analytics.

Data Lake Analytics

HDInsight involves creating a cluster. In case you want to run only parallel data analysis applications without worrying about clusters, then you can use Azure Data Lake Analytics service. Unlike HDInsight, a user here just specifies resource configuration required for running a query, and automatically those resources are provisioned and destroyed after processing is complete.

It supports a new language developed by Microsoft called as U-SQL which is combination of SQL and C# and can operate over structured and relational data.

AWS equivalent service – Kinesis Analytics.

Power BI

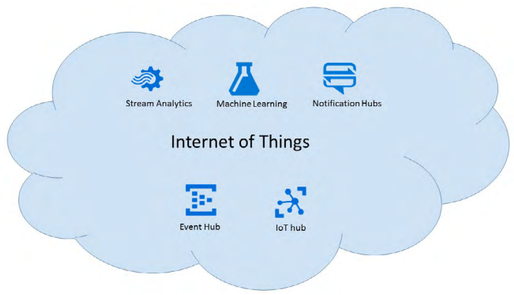
Power BI is a cloud based analytics service that builds visualizations, perform ad-hoc analysis and develop business insights from data.

The best part of Power BI is the rich support for integration with Azure services. To name a few, Power BI can read data and provide visualizations from various azure services such as Machine Learning, Stream Analytics, HDInsight, SQL Azure DB, Azure Storage, Event Hubs and so on.

AWS equivalent service – QuickSight.

Internet of Things (IoT)

Following is the diagram highlighting services present in the category of IoT -



Machine Learning (ML)

ML is also categorized as an IoT service and we have already covered it in the "Data & Storage" category.

Stream Analytics

The essence of streaming data is the speed at which it is generated, and if it is not analyzed immediately, then value of that data is lost. Example of real time data streaming are sensors on an oil well, fitness tracker on the wrist, toll payment devices in cars. These systems generate a lot of useful data and new data is produced constantly. This is the world we are living in, and IoT presents a number of opportunities for the same.

A traditional way can be to store the data and then process it at a later stage, but this may not be useful and can be too slow for many useful applications. So processing of continuous incoming data is required in such scenarios. This is where Stream Analytics can be used.

Azure Stream Analytics is an event processing engine. The event or data can be incoming from one stream or multiple streams like sensors, applications, devices, operational systems, websites and variety of other sources. A developer can use Stream Analytics Query Language which is a subset of T-SQL to issue queries on the incoming data. Each time new data arrives, the query runs and provides an output and this time duration window can be specified while creating the query in stream analytics. Hence Stream analytics queries on a slice of incoming stream rather than querying on the entire table of relational systems.

AWS equivalent service – Kinesis Analytics.

Notification Hub

Smartphones and tablets have the ability to "notify" users when an event has occurred. These are called as "Push Notifications". These notifications are pushed to devices even if the corresponding app is not active on the device, and this way the user engagement with app can be increased drastically. So "Push Notifications" is a vital component.

These notifications are delivered through platform specific infrastructure called "Platform Notification System (PNS)". For example, to send notifications to a windows store app, the developer must leverage "Windows notification service". To send notifications to iOS devices, the developer must leverage "Apple Push Notification Service (APNS)". These platform specific infrastructures do not offer support for broadcast, personalization and so on.

To avoid the headache of using different notification services depending on the platform, Azure Notification Hub provides a common interface along with other features to support push notifications across each platform.

AWS equivalent service – Simple Notification Service.

Event Hub

Event hub is an extension to traditional capabilities of service bus Queues and Topics.

The Service bus Queue and Topics use "Competing Consumers" model in which each consumer attempts to read from the same queue. This competition ultimately leads to scaling limits. On the other hand, Azure Event hubs works on "Partitioned Consumer Pattern" where each consumer reads only specific subset or partition. Using this strategy, Event Hubs adds massive scaling capability for ingesting data.
Working with streaming data commonly requires buffering of messages. Without buffering, you will miss the messages. Event hubs provides this buffering capability where message can be stored until they are processed.

I found an interesting and self-explanatory image highlighting difference between Event hubs and Service bus Queue/ Topics on Brent's (@BrentCodeMonkey) blog and found it apt to describe the difference.

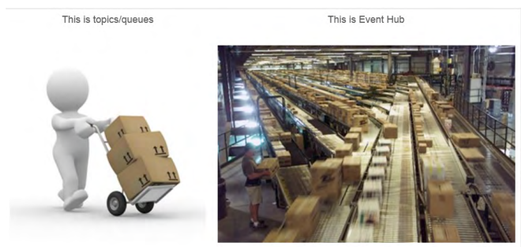


Image source: https://brentdacodemonkey.wordpress.com/2014/11/18/azures-new-event-hub/

So the major difference between topic/queue and event hub is of "Scale".

AWS equivalent service – Kinesis Firehose, Kinesis Streams.

IOT Hub

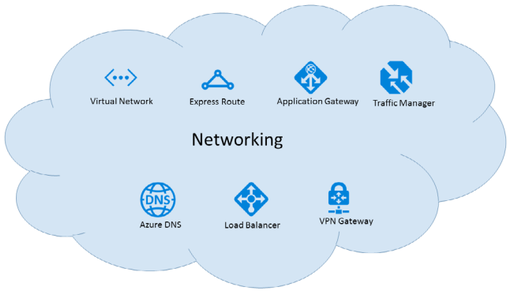
Stream analytics, HDInsight Storm, Spark streaming etc. services allows to create software that process streaming data. However none of them provide buffering capability which is a common scenario in case of IoT. Without buffering capability, data will be lost. To address this situation, Azure IoT hub service can be used.

IoT is based on Azure Event hubs and at the same time it allows features to talk back to devices generating data which is not possible with Event hubs.

Azure equivalent service – IoT.

Networking

Below is the list of services present under the networking category –



Virtual Network

Hybrid cloud implementation is a very common scenario. As a part of the hybrid cloud scenario, enterprises often look for hosting the front end of an application on Azure. However due to security

and compliance reasons, they wish to have the data stored on-premises. This is where connectivity between application front-end hosted on Azure, and Data store present on-premises is required. Azure VNET caters to such scenarios.

Using Azure VNET, you can control IP address blocks, DNS settings, security policies, route tables etc. within the network.

AWS equivalent service – Virtual Private Cloud.

Express Route

If we establish connectivity using Azure site to site, or Point to site, the VPN is established over public internet infrastructure only. This may introduce extra latency or low response time for end users which may not be acceptable in mission critical workloads. To overcome this situation, Azure offers Express Route service that can be used to establish VPN connectivity over dedicated private connection facilitated by the connectivity provider.

Express route connections do not go over the public internet as a result of which more reliability, faster speeds, lower latencies and higher security can be experienced.

AWS equivalent service – Direct Connect.

Application Gateway

Application Gateway on Azure is basically HTTP load balancer service that works on OSI layer seven load balancing semantics.

Http load balancer can be used to achieve –

1. Cookies based session affinity – Applications that require requests from same user/client session to reach the same backed virtual machine. Example is session maintainance in shopping cart application.

2. SSL termination/ offload – This way we can free web servers from SSL termination hence helping in improving the performance of the web server.

3. URL content based routing – Depending on the contents of the URL, you can route the request to server streaming videos, images etc.

AWS equivalent service – Elastic Load Balancing.

Traffic Manager

Traffic manager is a load balancer service on Azure that operates at DNS level. It has 3 strategies –

1. Performance – Let's say you have one application deployment in the US and another in the UK, then if users from the US region access the application URL, it is expected that the request should serve from the US region only as it will be the nearest region for a US based user. Similarly for the UK based user, the request should get served from UK region only for better response time. If you want this type of load balancing done automatically, then use "Performance" strategy.

2. Failover - Let's say you have one application deployment in US and another in UK. You want that the request coming from any location of the world should always get served from US deployment primarily. In case the US region deployment goes down, then UK based deployment should serve all the request. Once US deployment is up and working again, it should be treated as primary. If you want this type of load balancing done automatically, then use "Failover" strategy.

3. Round robin – This is a traditional round robin algorithm based load balancing; only difference being of DNS level.

AWS equivalent service – Route 53.

Azure DNS

Let's clear out a common misconception everyone has when they hear about this service – "It is NOT a domain registrar offering on Azure."

Azure DNS is a service that can be used for DNS delegation, so you still need to purchase a domain from a Domain registrar like Enom, HostGator etc. on top of which you can apply DNS delegation.

Azure DNS uses "Anycast networking" methodology which directs incoming request always to the nearest server. This helps to drastically improve the performance which may not be the case with non-delegated DNS. Plus inherent structure of Azure cloud platform offers high availability.

AWS equivalent service – Route 53.

Load Balancer

Traffic manager works at the DNS level, Application Gateway works at HTTP level, whereas Load Balancer works at the Network level.

For performing load balancing of traffic between virtual machines present in cloud service or VNET, Load balancer is used. For example, if we host a web application on Azure VM on port 80; then we can create load balancer for port 80 so that incoming request will be distributed across two machines in a round robin fashion.

AWS equivalent service – Elastic Load Balancer.

VPN Gateway

A VPN gateway is a type of networking device that connects two or more devices, or networks together, in a VPN infrastructure. It is designed to bridge the connection between networks or multiple VPNs together.

Azure VPN gateway serves the same purpose. It is used to send the network traffic between virtual networks and on-premises locations. It can also be used for sending traffic between VNET to VNET connections.

AWS equivalent service – VPN gateway is part of Virtual Private Cloud service.

Media and CDN

There are two services in this category as explained below –

Media services

If you wish to stream videos to HTML5, Flash, Silverlight, Windows 8/10, iPad, iPhone, Android, Xbox, Windows phone and any other client, irrespective of their streaming formats, then you need to build media solution that encodes and streams videos to various devices and client. Needless to state, it is a very complex task. Azure media services provides a cloud enabled solution for this problem.

Any video processing includes uploading, encoding

and protecting streaming and consumption steps. Azure media services provides numerous ways by which these processes become super easy for you.

AWS equivalent service – *Not available for all of the services of media solution. The details are as below –*

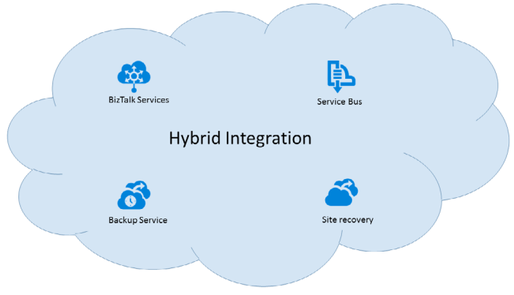
Azure Media Services	AWS
Encoding	Elastic Encoder
Lie and on-demand streaming	None
Media Player	None
Media Indexer	None
Content Protection	None

Content Delivery Network (CDN)

Azure CDN is a system of distributed servers that delivers content (mostly cached) upon request depending on the nearest geographic location of the request. In a way, this is similar to the caching mechanism but the difference in traditional caching and CDN is that, traditional caching works at application level, whereas CDN caches the data across various locations across the world.

AWS equivalent service – CloudFront.

Hybrid Integration



BizTalk Services

Biztalk can be provisioned on Azure either as IaaS (Biztalk installed on VM) or as PaaS (Biztalk services).

Biztalk service is a PaaS offering hence all types of maintenance is avoided, which is unavoidable with IaaS offering.

The Biztalk service main documentation page talks about Azure App Service and it looks like Azure Biztalk services will get replaced by App Services in the long run.

AWS equivalent service – None.

Service Bus

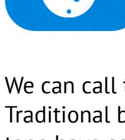
Applications often need to interact with other application and services. To facilitate this communication in the most powerful and easiest way so that decoupled system can communicate seamlessly with each other, Azure provides Service Bus offering.

The following services are provided by service bus –

Queue

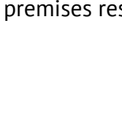
This is a brokered messaging service where messages are stored onto Azure service bus infrastructure so that both sender and receiver are not required to be always online for communication to happen. This is for one-to-one communication.

Topics-Subscription



This is similar to Queue but provides one-to-many communication capability.

Relay



This is used for point-to-point communication where sender and receiver both have to be always online. If you wish to access on-premises services without opening firewall or NAT, then service bus relay can be the option.

AWS equivalent service – for all – Simple Queue Service.

Backup Service

We can call this offering as "Backup as a service". Traditional backups taken on the physical storage or tape have scaling limitations. Azure backup service being a cloud offering offers unlimited possibilities plus you can introduce great level of automation in the backup tasks.

Backup service can be used to take backup of on-premises resources to Azure and restore back.

AWS equivalent service – none.

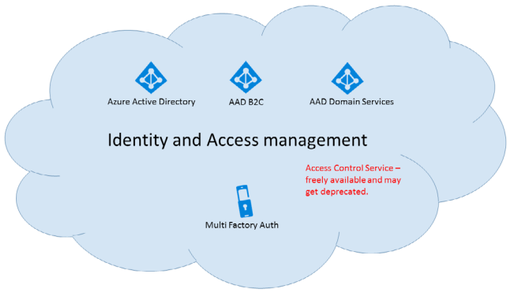
Site Recovery

We can call this service as "Disaster recovery as a service (DRaaS)". This service can be used to replicate VMs from on-premises environment such as VMware, HyperV directly to Azure instead of a secondary on-premises site.

AWS equivalent service – none.

Identity and Access management

Following services are present in this category:



Azure Active Directory

Azure AD is an "Identity Management Solution throughout the internet". Identity management is the only area today addressed by Azure AD. It is worth to mention Azure AD is not same as Windows Server AD.

Azure AD supports all modern protocols for authentication like WsFederation, SAML-P, OAuth and more in the future. This capability of Azure AD enables it to integrate with many enterprise level SaaS applications like Salesforce, Facebook at work,

and Twitter for business, DropBox and so on.

AWS equivalent service – Directory Service.

Azure AD B2C (Business to Customers)

This service is an identity management solution for consumer facing web and mobile applications. If you want your application users to be authenticated, then it is your responsibility to create an identity store database and then maintain it. If the requirement is to integrate this solution with social identity, then a change in the application is required depending on the API offered by various social integration platforms. To avoid this headache and simplify the integration with social identity of the user, AAD B2C can be used.

Using B2C, your users/ consumers can sign up using their existing social accounts such as (Facebook, Google, Amazon, LinkedIn) or can create new credentials which is termed as "Local accounts". AWS equivalent service – Directory Service.

AWS equivalent service – Directory Service.

AAD Domain Services (AADDS)

This can be referred as "Domain controller as a service" offering in Azure. As stated earlier, Azure AD is mainly an identity management solution and fundamentally it is not the same as Windows Server Active Directory. To bridge this gap eventually, Microsoft Azure has introduced AADDS. Using this service, you can join an Azure VM to domain without deploying Domain controller VMs which was not possible earlier.

AWS equivalent service – Directory Service.

Multi Factory Authentication

Today if you try to transfer funds or pay utility bills via internet banking, you are asked for Login credentials. After successful login, a verification code is sent to your handheld device (example, smartphone) in the form of a message. Once you have entered the code on the transaction site, the transaction is done. So in a way, the process of fund transfer is dependent on 2 step authentication. This is called as "Multi Factor Authentication".

Azure provides cloud enabled services using which you can implement MFA in your application very easily and securely.

AWS equivalent service – Directory service – multi factor authentication.

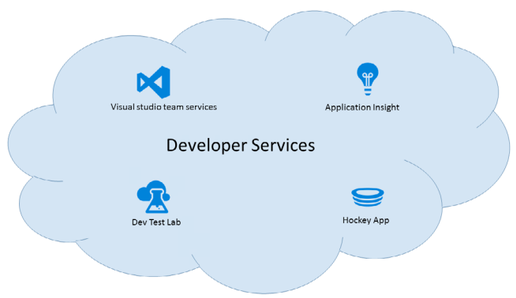
Azure Access control service

This service is made freely available on Azure and specifically designed for federated authentication implementation. However no update has been made to this service since a long time. Although there is no official word from MS about deprecation of ACS, at the same time there are no upgrades or SLA applications associated with this service.

The recommended approach is to use Azure AD based services and features.

Developer Services

The following services are present this category:



Visual Studio Team Services (VSTS)

Earlier VSTS was known as "Visual Studio Online". A common misconception around this was VSO allows you to write code online without the need to install Visual Studio on local machine, but this is not true. VSO or VSTS is Team foundation services on the cloud. So essentially it provides source control features over the cloud. Most of the features that were part of Team Foundation Server offering are now present on VSTS.

AWS equivalent service – None.

Application Insight

This service is an application monitoring service on Azure. Earlier App monitoring was dominated by tools from companies like AppDynamics, New Relic etc. With "Application Insight", Microsoft has entered into the space of "Application Performance Monitoring (APM)" tools.

This tool offers all usual services and features that may be required to understand the application health and performance parameters as a part of the monitoring process. It can help to diagnose and detect performance issues and what users are doing with your application.

AWS equivalent service – Cloud Watch, Cloud Trail.

Dev Test Lab

The usual challenges faced by Dev and test teams are delay in setting up a staging environment, creating replicas of production environment, Licensing issues, and cost. Dev Test labs allows you to create an environment in Azure with controlled cost options and with ease. Check out the next article by Vikram that talks about DevTest Lab in detail. The Dev Test Labs feature allows you to create Artifacts (a JSON template files) which contains instructions, commands to perform deployment and configuration. These artifact files are nothing but ARM templates which carries all ARM template benefits like create once-use always, idempotent deployment, focus on What instead of How for resource provisioning and so on.

AWS equivalent service – Device Farm.

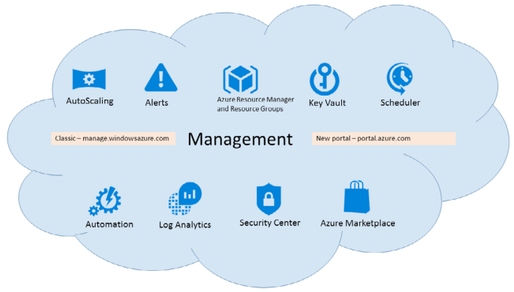
Hockey App

This service provides DevOps enablement for mobile applications. Hockey App is not integrated as a part of Application Insights. If you build mobile apps for any platform using any framework and language; using hockey app it becomes easy to invite users to test the app, collect feedback, metrics, live crash reports in Dev or production phase. This helps in making sure that users always get the latest updated version of the application and at the same time, developers get latest crash reports and feedback.

AWS equivalent service - None

Management

Following are the services which can be considered in this category:



AutoScaling

Autoscaling in Azure is supported for Cloud services roles, App Service plan and VM scale sets.

Scaling can be either vertical or horizontal.

Vertical – If you have a machine with 4GB RAM and you increase the RAM to 8GB for more computing power, then it is termed as Vertical scaling.

Horizontal - If you have a machine with 4GB RAM and you put another machine of the same configuration for parallel processing, then it is termed as horizontal scaling.

Azure supports both types of scaling.

AWS equivalent service - AutoScale

Alerts

Azure allows to configure alert notifications. These alerts can be for monitoring metrics or events on azure services. For configuration alert rule, a value is provided. This value is treated as threshold value and once it is crosses the limit, an alert is raised automatically. Alerts can be configured for application insights, sent to other systems, or can be set for billing notifications and so on.

AWS equivalent service – Cloud Watch, Alarms

Key Vault

Key Vault allows developers to implement cryptographic functionality in an application without worrying about storing and managing associated keys. Key Vault can be used to encrypt keys and secrets (like storage account key, connection strings) using the keys stored on Azure Key Vault and protected by Hardware Security Modules (HSM). These keys can also be used to encrypt sensitive data directly.

AWS equivalent service – Key Management Service, Cloud HSM.

Scheduler

Scheduler can be used for scheduling job invoking http/s endpoints, making entry in queue storage or any services that are inside or outside of azure. For example, you may want to ping your website address after every 1 minute to check if status code is 200 (ok) and there is no error. This can be one of the real life implementations of scheduler job. Please note that scheduler will only schedule the job and not execute it.

AWS equivalent service – None.

Automation

Azure Automation is "Configuration Management Service" on Azure. Popular services in this area are **Chef** and **Puppet**. This provides a way for users to automate manual, long running or frequently repeated tasks. For example, you are using a VM on Azure and during non-business hours, you want the VM to shut down every day and start every morning, then you can use Azure Automation.

AWS equivalent service – OpsWork, Cloud Formation.

Log Analytics

Log Analytics is the service in Operational Management Suite (OMS) that helps you gain insights into details of infrastructure hosted on premises and cloud.

App insights is for monitoring applications telemetry, whereas Log Analytics is for capturing telemetry of infrastructure components.

AWS equivalent service – CloudWatch, Cloud Trail

Security Center

This service provides a holistic view of security state of all the azure resources owned by you. Once you enable "data Collection" for security center, then the data that gets collected automatically gets analyzed for security incidents and notifications, alerts are displayed about possible vulnerabilities and recommendations.

AWS equivalent service – Inspector.

Azure Marketplace

Azure Marketplace is a great platform where you can globally promote your products built on top of Azure services. Without a marketplace, it is almost an impossible and expensive task if you decide to do promotions on your own. Marketplace today is available in around 56+ countries hence making it possible to sell your product in those countries easily and giving you a global reach. Plus the billing part is embedded in Azure billing which makes it an even more attractive solution to market your products. You can publish VM based solutions, Machine learning web services, API App connectors, web applications as SaaS on Marketplace and monetize them.

AWS equivalent service – AWS Marketplace.

Azure Portals

Today there are two portals for Azure –

1. Classic portal – http://manage.windowsazure.com

2. New portal – http://portal.azure.com

The future lies in the new portal which provides more options and better control over azure resources.

Miscellaneous

Here I am listing those Azure service and their AWS equivalents which I could not fit in any category but find it worth mentioning them.

Azure	AWS	Remarks (if any)
Azure CLI Azure PowerShell	Command Line Interface	For management of resources through CLI or PowerShell. Provides another way to manage resources than UI portal.
Resource Manager	Cloud Formation	User for grouping of resources based on type, intent, usage.
Xamarin Test Cloud	Device Farm	Range of services for front device testing and simulation.
Azure Batch		Used for job based applications
Availability Sets		Used for obtaining high availability for VMs hosted in Azure
Azure SDK		For accessing Azure services through various programming languages and platforms.
Cognitive Services		Provides APIs for Language, speech, face recognition etc. so as to enable developers to build powerful intelligence in applications.
WebJob		Background processing component for Azure App service.

Where can I get the Azure service Icons?

Most of the times, we need to draw architectural diagrams while presenting Azure services to clients in Proposals and pre-sales activities. In such situations, Icons representing Azure services will come very handy and it will make the Architecture diagram look more self-explanatory; as you would be using real world Icons instead of using generic squares or circles and annotating them.

Now the question is, "how can I get these"? Well get them here - https://www.microsoft.com/en-in/download/details.aspx?id=41937. This URL has all Microsoft product Icons, PNGs, PPTs, Visio diagrams and so on and is a great essential link for Documentation.

An Appeal to readers

Thanks for reading through! I hope you found this guide useful. Azure services are being developed very actively and in reference to AWS to Azure mapping, things can change. While we will keep an eye out, I would request you to let us know from time to time when you comes across a change, a new service replacing old one and similar.

Conclusion

This article enlightens how Microsoft Azure cloud platform is becoming richer day by day and almost everything is going to be cloud oriented very soon ■

About the Author

kunal chandratre

Microsoft® MVP Most Valuable Professional

Kunal Chandratre is a Microsoft Azure MVP and works as an Azure Architect in a leading software company in (Pune) India. He is also an Azure Consultant to various organizations across the globe for Azure support and provides quick start trainings on Azure to corporates and individuals on weekends. He regularly blogs about his Azure experience and is a very active member in various Microsoft Communities and also participates as a 'Speaker' in many events. You can follow him on Twitter at: @kunalchandratre or subscribe to his blog at http://sanganakauthority.blogspot.com

Vikram Pendse

DEVTEST LABS IN MICROSOFT AZURE

Microsoft Azure is a Cloud Platform offering various flavors of Services like Platform as a Service (PaaS), Infrastructure as a Service (IaaS) and Software as a Service (SaaS). Many new features are added to the Azure platform on a continuous basis to enable businesses to move their workloads on the cloud and speed up overall development and delivery cycles. **Overall for any Cloud application development, creating Infrastructure is the key component**. Usually it takes a lot of time, efforts and cost to build a development and testing environment on-premises and maintain it for a longer duration.

In this article, we will explore the "DevTest Labs" offering and how it enables businesses to quickly roll the Development and Test environments on Azure with value added features like Cost Monitoring, Resource Management, Automation and control of environment.

Current DevTest Scenario On-Premises

Currently most organizations have their DevTest environments configured On-premises and usually these are complex in nature. We explicitly need to maintain and monitor the infrastructure. We also face downtimes or blockages if there are issues in networks or in physical boxes. We also need to be dependent on our System Admins/IT Admins for the entire DevTest environment setup to run smoothly. If we try to build a 'Virtual' DevTest environment On-premises (Not on Azure or any cloud but using In-House Datacenters), more or less we face the same issues:

1. Delays in environment provisioning

2. Maintenance of the environment (Updates, Patches and overall maintenance)

3. Cost and Efforts to provision and maintain the infrastructure

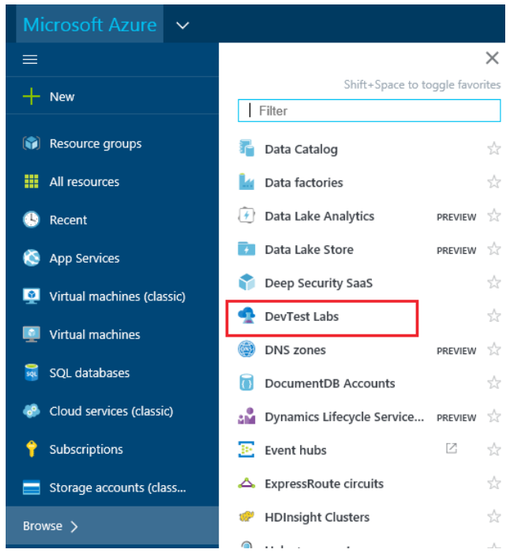
4. Dependency on IT Administrator/System Administrator and IT Team

Introduction to DevTest Labs in Azure

Azure DevTest Labs offering introduced last year and recently announced as GA (General Availability), solves most of the issues and challenges which businesses face in their actual On-Premises or Virtual Environments (In House Datacenters). In a nutshell, the DevTest Lab is kind of a self-service sandbox environment in Azure.

Setting up the DevTest Labs

Firstly, you need a valid Microsoft Azure subscription. Please note that you can setup and access DevTest Labs from the new Azure Portal (Ibiza) at https://portal.azure.com . You can click on Browse > DevTest Labs as shown below



Once you click on DevTest Labs, it will ask you for five Parameters such as:

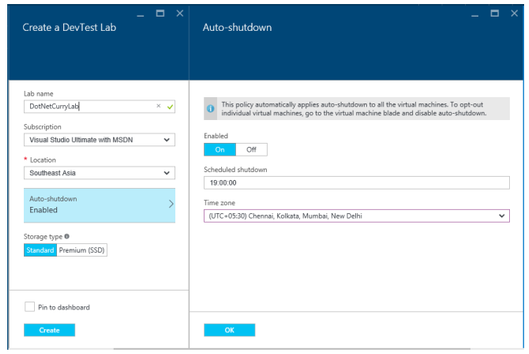
1. Lab Name – Any generic name which you / your project wish to keep

2. Subscription – You can choose any subscription if you have multiple ones. Else it will be a default subscription.

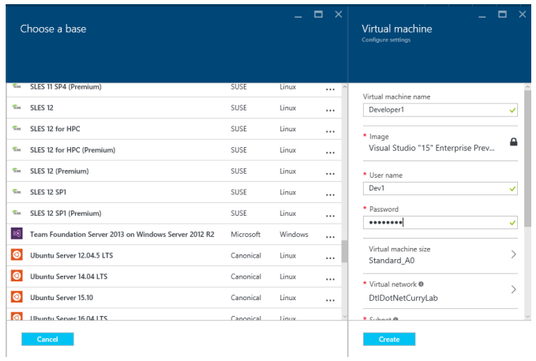
3. Location – Nearest Datacenter region where you want to create this environment. You need to choose nearest Datacenter to avoid latency issues

4. Auto-shutdown – This feature allows you to Shutdown your environment at a given specific time based on the selection of a time zone. There is an another "Auto-Start" option as well to turn ON the environment which we will discuss later in this article.

5. Storage – Storage is the backbone of any Azure service whether it is PaaS or IaaS. We need to define storage as all VM's VHDs will be stored along with other data in Azure storage. Based on your requirement, you can choose either Standard or go for Premium Storage (Premium SSD)

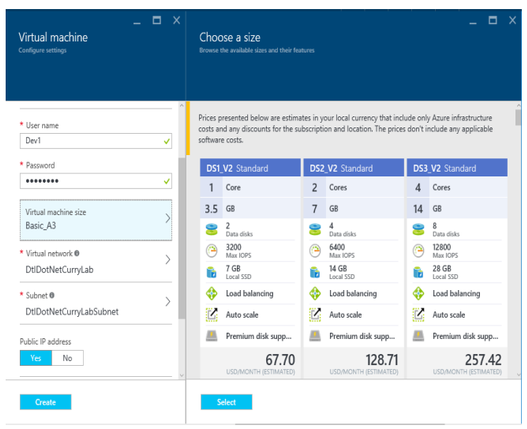


This will now create a structure for your DevTest Lab. After this you need to start adding VMs as per your requirement. Kindly note that by default the deployment model for VMs will be Azure Resource Manager (ARM) and not Classic (Cloud Services). Azure DevTest Labs does not restrict you to create only specific Windows VMs, but it offers a variety of Images in the Gallery including Linux VMs. To a large extent, the VMs which you create from Virtual Machine section, is the same that you can see here.



Creating Virtual Machine in DevTest Labs

You can select any of the Virtual Machine Images from a set of Base images. Note that all these images are provided by Azure and it does not give you the option to choose Custom Images. However, custom images of the Virtual Machines can be prepared in the DevTest Labs, post provisioning. You can see the option for the same as "Create Custom Image (VHD)" under VM > Settings

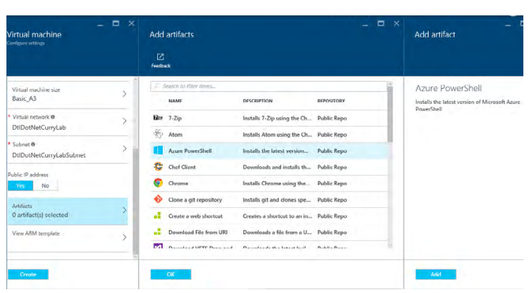


For VM creation, the steps are very simple and you need to give minimum required information like Username, Password and Size of the VM. Note that it also automatically manages VNET and Subnet for you. You will also get option to Turn ON or OFF the Public IP of the individual VM during the provision.

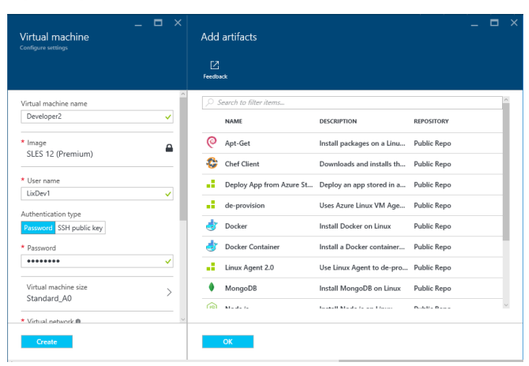
Adding Artifacts to Virtual Machines in DevTest Labs

One of the common task for all IT Administrators or for that matter Developers too is that we need to install some common set of tools and packages which we use in our day-to-day development along with Visual Studio (you can already get Visual Studio based image in VM Image gallery, so this solves the problem of installing VS on VM). "Artifacts" ease out the overall process for you here. Since by default Azure VMs are connected to internet, it becomes quite easy and seamless to download packages on VM and install them. "Artifacts" section is as big and rich as the VM Gallery and offers a variety of basic and standard packages which you need for your Development and Testing work on the environments. (For example, 7-Zip, Azure PowerShell, Google Chrome etc.) The most amazing part of "Artifacts" is, it changes the list of available packages for you based on your choice of VMs i.e. Windows or Linux. So irrespective of the VM you choose, you still get all the benefits of "Artifacts" as shown here:

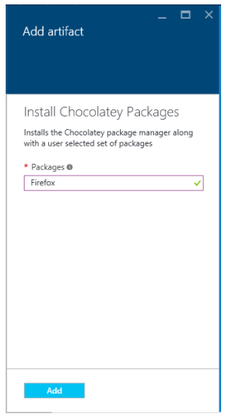
For Windows:



For Linux:



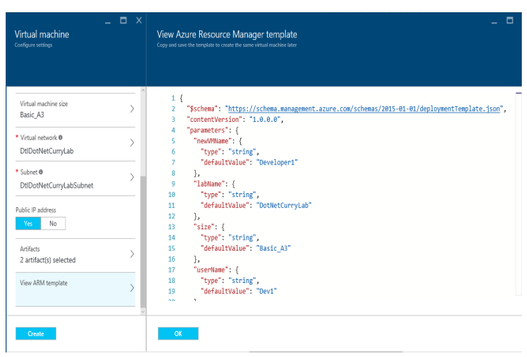
If you think that the package you are looking for is not in the list provided by "Artifacts" gallery, then Azure also gives you the flexibility and freedom to associate a package name. For example, you can get package name from the "chocolatey" repository and give/associate the package name here



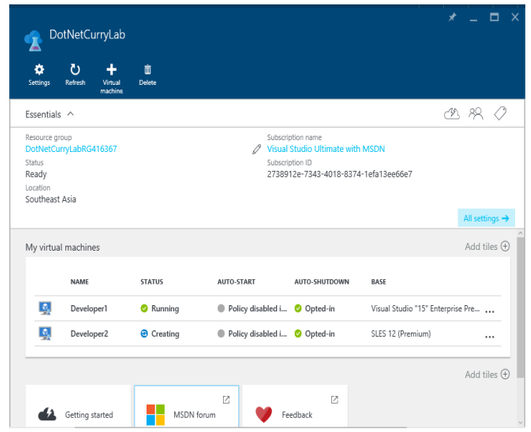
There are around 4000+ community packages maintained. You can visit them here: https://chocolatey.org/packages

Azure Resource Manager (ARM) Template of individual VM

While provisioning the VM in DevTest Labs, Azure provides you an Azure Resource Manager (ARM) template which you can save and use at any point of time if you wish to provision the VM with same settings and specifications. ARM Template is available across multiple components in Azure which you might be using already.



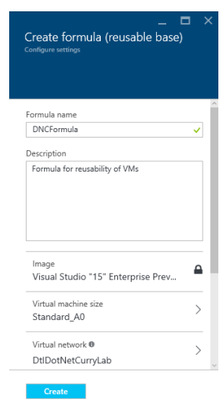
Finally, after all these setups of VMs you can see the list of all provisioned and available VMs on the dashboard as shown here:



Additional Value-added features of DevTest Labs

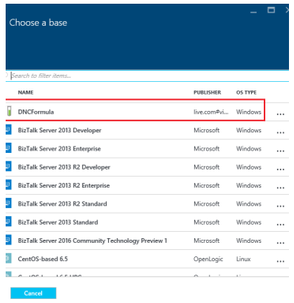
Creating Formula (reusable bases)

This is a mechanism to ease and replicate (not fully automated) the VMs in your DevTest Labs. Suppose you are required to create 5 VMs with similar specifications, then "Formula" allows you to save a common set of specifications, which you can save, and can use multiple times for the other four VMs, instead of doing it manually one by one.



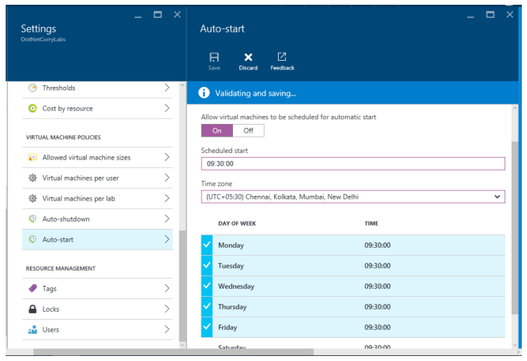
You can also "Update" the formula at any point

of time as per your requirement. You can notice that the Formula is visible post creation in the VM Gallery, and you can see the same when you start picking up the new base for your new VM from the gallery.



Auto Start and Auto Shutdown of VMs

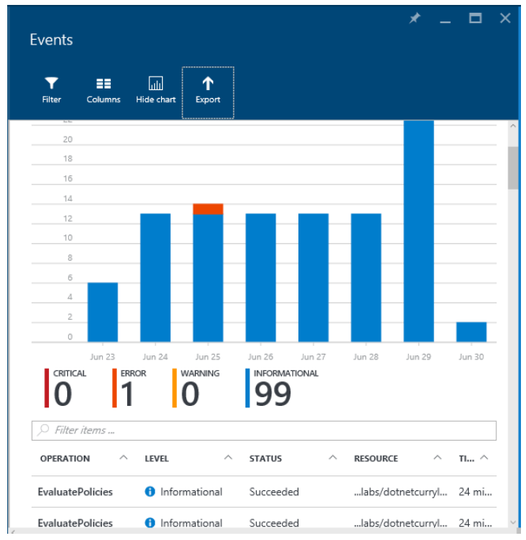
We have already discussed the Auto Shutdown of VMs which Azure asks us while creating the template for DevTest Labs. You can make changes to the same, post provision as well. The Auto Start option allows you to control your VMs "ON" timings based on your choice of time and time zones for a particular day or the entire week.



This feature was one of the most requested ones for granular control of the environment so that Azure customers can Turn ON and Turn OFF the environment at any point of time. The main reason is to put control on the cost and usage. Previously customers and enterprises were using various mechanisms like Scheduler, Automation Scripts / Automation jobs or some tools based on REST API to Turn ON and OFF VMs. In DevTest Labs, you no longer need these workarounds or additional efforts to perform Auto Start and Shutdown of your VMs, it does out of the box once you configure it as per your need.

Audit Logs

You can see Audit Logs of the entire DevTest Labs components with necessary details along with graphical representation. Like other Audit Logs of various Azure components, you can use them for troubleshooting and getting more information about events. During support tickets or support activity, Audit Logs are always helpful. You can access them from Settings > Audit Logs



You can also Export your Audit logs and archive them in the storage or stream them to Azure Event Hub, but this Export feature is currently in PREVIEW mode.

Virtual Machines per Lab and per User

In order to utilize the environment in an optimized way, creators or administrators of Labs can define the maximum number of Virtual Machines allowed per Lab and maximum number of Virtual Machines allowed per User as well. This allows you to setup a limit on the number of VMs per lab and number of VMs per User.

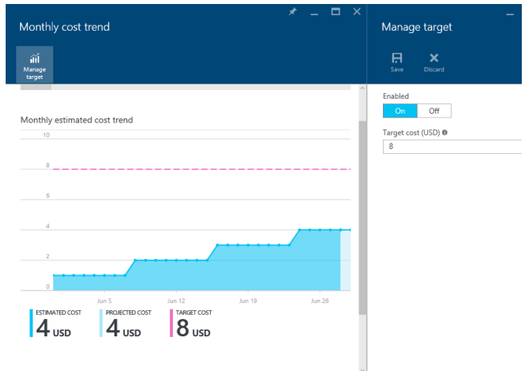
Resource Management – Tags and Users

As we discussed earlier, the entire deployment strategy for DevTest Labs is Azure Resource Manager (ARM). Hence some of the features which comes with ARM by default, are also available and applicable for DevTest Labs. The Best example is adding Tags in Key-value pair format and providing restricted access to Users or do a user management via Roles (RBAC – Role based Access Control). These features ensure individual responsibility of the DevTest Labs components by Tag and puts restrictions on usage via User management.

Monthly cost trend and Cost by resource

Since it is ARM based and you can have multiple components in the DevTest Labs, hence it allows you to see the per resource consumption. For this you need to go to Settings > Cost by resource.

Monthly cost trend allows you to Manage Target or put some Target on the consumption. This is kind of a projection and is helpful from a better cost and resource consumption point of view.



Those customers or businesses who are currently evaluating Azure and have trail Azure accounts with limited credits will find this feature very useful.

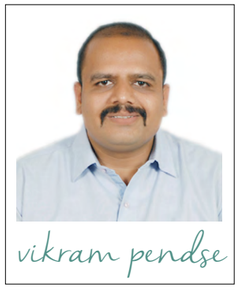
What is the Pricing Model of DevTest Labs in Azure?

Azure DevTest Lab is a FREE Service. The overall pay-as-you-go model charges you for the usage of Azure resources like VM and storage. VM pricing is based on the type and size of VM you choose and standard VM Pricing is applicable. Hence besides this, there are no additional charges to be paid for DevTest Labs. For more pricing related information, you can visit https://azure.microsoft.com/en-in/pricing/details/devtest-lab/

Conclusion:

Today's era is of Rapid Application Development and these applications and services need hosting on a robust, scalable and highly available platform. All businesses and customers have a common requirement of creating DevTest environments for their development or project work for a short span of time, or for a longer duration. Until now, they would spend a lot of time, efforts and resources to create these virtual DevTest environments. However now, Microsoft has addressed this issue by making DevTest Labs service available on Azure. Azure DevTest Labs solves the problem with customization and automation it provides. Now you can create DevTest environments within few minutes even if you don't have a deep knowledge of automation and complex PowerShell scripts or REST APIs for that matter ■

About the Author



Vikram Pendse is currently working as a Technology Manager for Microsoft Technologies and Cloud in e-Zest Solutions Ltd. in (Pune) India. He is responsible for Building strategy for moving Amazon AWS workloads to Azure, Providing Estimates, Architecture, Supporting RFPs and Deals. He is Microsoft MVP since year 2008 and currently a Microsoft Azure and Windows Platform Development MVP. He also provides quick start trainings on Azure to startups and colleges on weekends. He is a very active member in various Microsoft Communities and participates as a 'Speaker' in many events. You can follow him on Twitter at: @VikramPendse

THANK YOU

FOR THE 4th ANNIVERSARY EDITION

@yacoubmassad

@kunalchandratre

@sravi_kiran

@ subodhsohoni

@damirarh

@ vikrampendse

@shobankr

@suprotimagarwal

@gilfink

@saffronstroke

WRITE FOR US