

XPR.T.

Magazine N° 5/2017

Discover endless possibilities

A Reactive Cloud: Azure Event Grid

Scaling Scrum to the limit

Deployment pipelines for ARM

Containerized Testing



PROUDLY PART OF XEBIA GROUP

Think ahead. Act now.



~~UNMANAGEABLE~~

Reclaim your infinite ability to ADAPT and DELIVER

We proudly presented TechDays, so when can we present your company to your clients, partners or even your own employees? Our dedicated, passionate, and creative team are here to take your events to the next level.

GP Strategies® Netherlands is a leader in B2B event management and marketing services.

Connect with us today at gpnetherlands@gpstrategies.com



Colofon

XPRT. Magazine N° 5/2017

Editorial Office

Xpirit Netherlands BV

This magazine was made by

Vivian Andringa, Pascal Naber,
René van Osnabrugge,
Martijn van der Sijde, Loek Duys,
Alex Thissen, Kees Verhaar,
Geert van der Cruisen,
Chris van Sluysveld, Marco Mansi,
Cornell Knulst, Marcel de Vries,
Pascal Greuter, Alex de Groot,
Roy Cornelissen, Jesse Houwing,
Sander Aernouts, Jasper Gilhuis,
Marc Duiker, Peter Groenewegen

Contact

Xpirit Netherlands BV
Laapersveld 27
1213 VB Hilversum
The Netherlands
+31 (0)35 538 19 21
pgreuter@xpirit.com
www.xpirit.com

Layout and Design

Studio OOM
www.studio-oom.nl

Translations

TechText

© Xpirit, All Right Reserved

Xpirit recognizes knowledge
exchange as prerequisite for
innovation. When in need
of support for sharing,
please contact Xpirit.

All Trademarks are property of
their respective owners.

Gold

Microsoft Partner



docker
PARTNER

If you prefer the
digital version of
this magazine,
please scan the
qr-code.



In this issue of **XPRT.** Magazine our
experts share their knowledge about
discovering endless possibilities



INTRO

004 Communities to enable
endless possibilities

FUTURE TECH

009 A Reactive Cloud:
Azure Event Grid

014 From Search to
Checkout without annoying
your customers

DEVOPS ADVENTURES

018 Scaling Scrum to the limit

023 8 Years of CPU in a Day

CLOUD STRATEGY

027 Cloud Transitions
done right!

032 Containerized Testing

038 IaaS, Containers or
Serverless?

INFRASTRUCTURE AS CODE

042 Containers as a
Service in Azure

046 Best practices using
Azure Resource Manager
templates

055 Deployment pipelines
for versioned Azure Resource
Manager template deployments

Communities to enable endless possibilities

Last year, Linux OS celebrated its 25th anniversary. In these years, Linux - and a lot of other Open Source Software (OSS) - has changed the world. While the concept of Open Source Software has made a significant difference to the industry, there is another driving force underneath successful OSS software: its community. While approaching the year 2018 at full speed, at the same time communities has emerged around us. So before you dive into the depths of the latest and greatest Microsoft technologies in this issue of XPRT, we'd like to elaborate on the power of Communities.

Author Jasper Gilhuis & Alex de Groot



Emergence of communities

In the early days of both the Greek and Roman empires, it was quite common for wise men to gather together and share their thoughts. The entire philosophy of Democracy is based on these gatherings. Therefore, it's not strange that the word community finds its origin in the Latin word 'communis', meaning 'shared in common'.

With the growth of IT in the second half of the 20th century, gatherings found a central place in the IT business. End User Groups were founded to share experiences, IT Pro's shared their knowledge on mailing lists, and many developers became active on online forums. Soon this drew the attention of researchers, which resulted in studies that correlated community participation to increased productivity.

Nowadays, we see communities embedded in both our social and professional lives. Social networks, WhatsApp groups, Slack, GitHub and StackOverflow are known to everyone. Community principles became deeply embedded into project guidelines such as the Agile manifesto. The low-entry structures of communities with respect for everyone's opinion are here to stay.

Technology 'Ecosystems'

Compared to other industries, communities are probably most deeply embedded in IT. While often gathered around a product or field of interest in other industries, in IT they are usually related to an area of expertise. Engineers have the tendency to focus their complete attention on a single technology. We see this in the structure of the community. Completely focused "ecosystems" on a single technology allow deliberate but limited learning, and are difficult to open up to adopt new ideas.

A good example of such an "ecosystem" is the Open Source .NET Community. After Microsoft's first source drop of the .NET code late 2014, they started to develop the platform in close collaboration with the community. Using open forums, ticketing systems, podcasts, and even open design sessions, they allowed the community to participate in the overall design and source code of .NET. The product of the community is very tangible: Core and derived community components have never been in better alignment with the design concepts of .NET.



"Alex will make the impossible possible with his smarts and endless perseverance."

Marcel de Vries about
Alex de Groot



"Technologies were clustered into several distinct "ecosystems" that tended to be used by the same developers. On the left of this chart we can see a large cluster representing web development (with JavaScript at the center) and one for Microsoft technologies (centered around C# and Visual Studio). On the right we see a constellation connecting Java, Android, and iOS. Other smaller correlated clusters included C/C++/Assembly, Raspberry Pi with Arduino, and languages like Python and R alongside language-specific IDEs."

Source: <https://xpirt.it/xprt5-com1>



Introducing yourself to such an "ecosystem" can really enrich your knowledge and can help you gather new ideas. Not only is learning about the technology a fun experience, but it also increases your employability!

Community engagement is – like volunteer work – highly rewarded by potential employers. Stepping outside of your familiar communities or even your comfort zone can bring about a whole new experience!

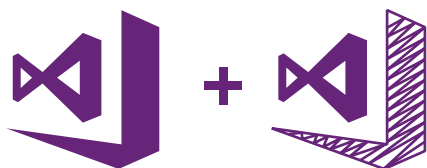
Community-based Product Development

We already mentioned that .NET and especially .NET Core have benefitted tremendously from an active community. The community helps to

shape new ideas long before they even get embedded into a released product. This form of co-creation is helpful because it allows early experiments before a single byte of production code is produced.











Co-creation hits the heart of the influence of the community on product development. While the feedback on new ideas flies in almost instantly, feedback on existing offerings helps product owners set priorities. It also allows them to identify complicated or misunderstood areas in their products.

Microsoft, Google, Facebook, and all other tech giants use this form of co-creation all the time. How many times haven't you tried Gmail Beta or even an Alpha of ReactJS? These days, Microsoft even issues Go-Live licenses for Visual Studio and TFS pre-releases. Why? Because the burden of supporting pre-released software is cheaper than releasing the wrong thing.



Participate in the community! How? Download Visual Studio Preview now! It runs side by side with your other installations.

<https://www.visualstudio.com/vs/preview/>

Organizations with the most open source contributors		
	Microsoft	16,419
	facebook	15,682
	docker	14,059
	angular	12,841
	google	12,140
	atom	9,698
	FortAwesome	9,617
	elastic	7,220
	Apache	6,999
	npm	6,815

Microsoft ♥ Communities & OSS

As part of their global strategy, Microsoft has taken a major leap in embracing Communities and OSS. In the last few years, they have overcome the historical sentimental gap between themselves and the OSS world. The most tangible examples of these are highlighted below:

- > SQL Server 2017 will be released on Windows, Linux and Docker
- > Azure Cosmos DB can be used from any NoSQL adapter
- > Bash runs on Windows¹
- > Visual Studio is now fully featured on macOS.

Microsoft's commitment to the community goes so far that it appears that all their activity is now on Github.

In last year's statistics, they became the most active OSS contributor! These numbers illustrate that it is beneficial for a company to not only rely on open source software, but also contribute to it. Have a look for yourself at <https://github.com/Microsoft>. Community features have even been embedded into Microsoft's product offering. Microsoft Visual Studio Team Services (VSTS), a Microsoft Azure Cloud-based SAAS offering, enables teams to work closely together on software development projects while providing a rich tool set and capabilities that can be used across platforms. This includes a wide range of topics such as integration with Maven, Jenkins, NPM Package Management, and many other cross-platform developments. In addition, advanced pull request workflows and Git forking enables communities inside companies to leverage the power of the collaboration. While this is known to many Microsoft developers, it is less known by other communities. VSTS could easily act as bridge between these communities!

Rules of Community Engagement

Participating in communities is free of charge but does not come for free. You can't only take something from the community. Every participant is expected to make their contribution to the community. This can be as simple as thanking community members for their efforts in answering your



"During my early years as a consultant, I spent my spare time answering questions on mailing lists. It helped me understand and take on challenges from different angles. Later, it allowed me to quickly identify common errors and extract best practices."

Xpirit's managing director, Pascal Greuter

¹ <https://xpir.it/xprt5-com2>

² <https://xpir.it/xprt5-com3>

question. Another way is voting up answers on StackOverflow. The most important unwritten rule is to pay back everybody's willingness to help with kindness and respect.

Some communities have strict Contribution Policies. A great example of this is the .NET Roslyn contribution policy². It states clearly what is expected from you when you want to add something to the source code of the Just-in-Time compilation component of .NET.

You also need to understand that the community gives you a good answer or an excellent library, but you shouldn't trust it blindly as if it's your own. Microsoft has recently published a blog-post on their DevOps blog, explaining

that many Open Source projects contain vulnerabilities³. You'll always need to take any piece of input, use it to the best, but make it your own and add your own craftsmanship on top of it.

Conclusion

Communities deliver an awesome foundation for the modern engineer. They give you answers, deliver a learning environment, and even building blocks to accelerate. We at Xpirit try to leverage communities wherever possible. Every time we use something, we try to give back something. Feedback, thumbs up, thumbs down, and pull requests on GitHub. The XPRT-magazine is another example which helps us to combine what we learn and deliver it back to you.

We encourage you to become a community member. Get active! Don't be shy, there's plenty of people out there who have made mistakes. And the community is forgiving. They won't even bother about your mistakes, they will love it. Because other community members will learn from things you experience as difficult. Try and share the love! Tweet to us (@xpiritbv) with your greatest community stories and we'll reward you for the community love. It will surely enable you to achieve endless possibilities! `</>`

³ <https://xpir.it/xprt5-com4>



"Xpirit's personal teddybear. Always there to help people out with a cuddle or by optimizing their ALM process."

Geert van der Cruijssen about Jasper Gilhuis

A Reactive Cloud: Azure Event Grid

Serverless compute or Function-as-a-Service (FaaS) is still gaining a lot of traction in the software industry due to the reduced time to market, lower operational and development costs, and ease of scalability. The Azure platform already provides Azure Functions and Azure Logic Apps, two serverless services to build modern, event-based, reactive systems. Last August Microsoft extended its serverless offering by introducing Azure Event Grid, a fully managed event routing service, capable of managing the routing of millions of events per second from any source to any endpoint, the IFTTT¹ for the Azure platform.

Authors Marc Duiker & Marco Mansi

Designing modern event-driven systems

Any modern business-critical system must be:

- › **Responsive:** The system responds in a timely manner, which ensures usability, utility and consistent behavior of the system.
- › **Resilient:** The system stays responsive when failures occur. This can be realized by isolating components of the system so they can fail and recover without failing the entire system.
- › **Elastic:** The system stays responsive during varying workloads. This can be realized by increasing and decreasing the resources for individual components without causing bottlenecks anywhere in the system.

These are exactly the features which describe Reactive Systems (<http://www.reactivemanifesto.org/>). These systems are based on a message-driven (asynchronous) architecture, so they are loosely-coupled. This makes them easier to change and they are more tolerant to failure than non-message-based architectures.

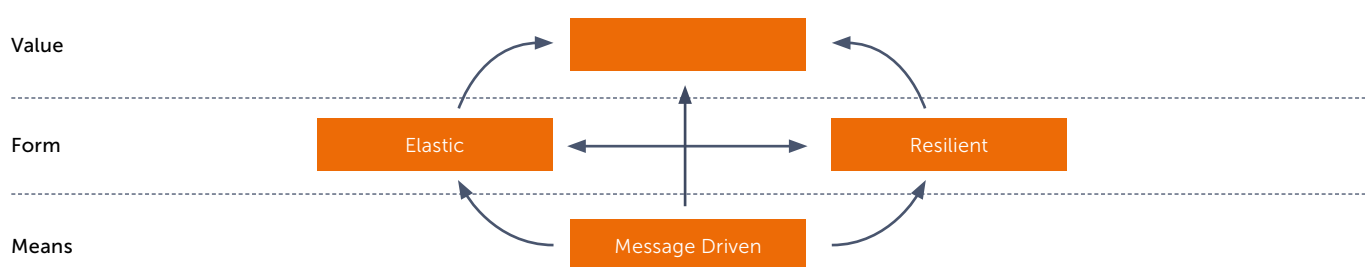


Figure 1: Reactive Systems

With Azure Service Bus, Microsoft was already providing a service for message-driven systems based on either Queues (for one-to-one messaging) or Topics (one-to-many messaging). Service Bus is a very powerful but also rather complex messaging solution, particularly suitable for enterprise solutions that require transactions or duplicate detection. Let's see what Azure Event Grid has to offer.

¹ If This Then That - <https://ifttt.com/>

Azure Event Grid

The new Event Grid is a more lightweight event service that is highly integrated within the Azure platform, which means that it can be easily configured to work with Azure Functions, Logic Apps, and many other Azure services. By using custom topics and webhooks, Event Grid can even integrate with applications outside Azure, so it is an ideal candidate for cross-service and cross-cloud scenarios. With Azure Event Grid, Microsoft now offers the technology for a push-based, message-driven architecture that allows developers to create reactive systems in a serverless application landscape.

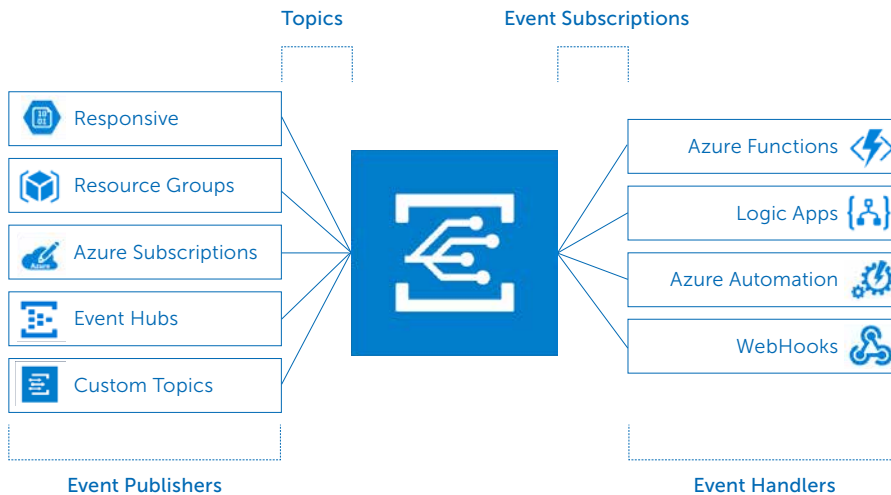


Figure 2: Azure Event Grid

Azure Event Grid Concepts

Let's explain Event Grid by means of an example:

For a given Azure subscription we want to verify the storage accounts created in that subscription. The verification is just an email that is sent to an administrator using Logic Apps.

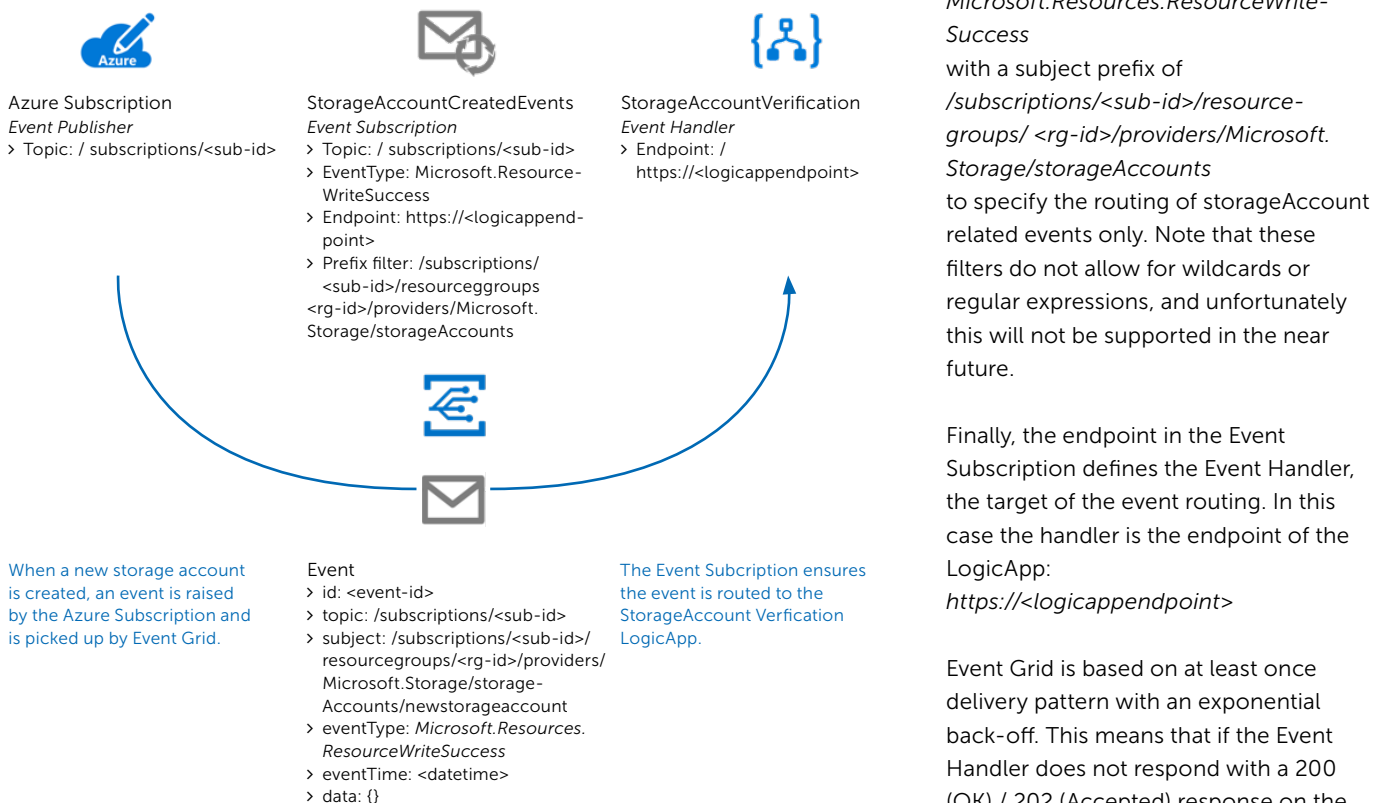


Figure 3: Azure Event Grid Example

In this case, Azure Subscriptions is the Event Publisher. The publisher categorizes events into topics. The topic (/subscriptions/<sub-id>) is a system topic since it is provided by Azure. It is also possible to create custom topics and events that can be raised from a custom application or function. Note that the maximum size of an event is 64KB.


Every time a resource within the Azure subscription is created successfully, an Event of type Microsoft.Resources.ResourceWriteSuccess is raised and published to the topic endpoint.

The event is then picked up by Event Grid and processed on the basis of an Event Subscription. The Event Subscription defines the routing of events from a certain topic to an Event Handler. The routing can be based on the EventType and prefix & suffix filters on the event subject.

In this case we created a subscription called *StorageAccountCreatedEvents* that points to the topic: /subscriptions/<sub-id> where we only want to handle events of type *Microsoft.Resources.ResourceWriteSuccess* with a subject prefix of /subscriptions/<sub-id>/resourcegroups/<rg-id>/providers/Microsoft.Storage/storageAccounts to specify the routing of storageAccount related events only. Note that these filters do not allow for wildcards or regular expressions, and unfortunately this will not be supported in the near future.

Finally, the endpoint in the Event Subscription defines the Event Handler, the target of the event routing. In this case the handler is the endpoint of the LogicApp: https://<logicappendpoint>

Event Grid is based on at least once delivery pattern with an exponential back-off. This means that if the Event Handler does not respond with a 200 (OK) / 202 (Accepted) response on the request, Event Grid will retry, using



"Creates Azure solutions like a true Italian: with lots of passion and aiming for Ferrari-like perfection."

Martijn van der Sijde about Marco Mansi

increasingly longer intervals (up to one hour) and will stop retrying after 24 hours. The message will be dropped if it can't be delivered. A full description of the Event Grid message delivery and retry mechanism can be found at:

<https://docs.microsoft.com/en-us/azure/event-grid/delivery-and-retry>

Event publishers

At the time of writing this article, Azure Event Grid is in preview and supports the following Event Publishers: Azure Resource Groups, Azure Subscriptions, Blob Storage², Event Hubs, and Custom (Event Grid) Topics.

These are the currently available Event Types that can trigger events. Please note that by using Custom Topics, events can be triggered from custom applications that can be hosted anywhere.

Event Publisher	Event Type
Azure Resource Groups & Azure Subscriptions	Microsoft.Resources.ResourceWriteSuccess
	Microsoft.Resources.ResourceWriteFailure
	Microsoft.Resources.ResourceWriteCancel
	Microsoft.Resources.ResourceDeleteSuccess
	Microsoft.Resources.ResourceDeleteFailure
	Microsoft.Resources.ResourceDeleteCancel
Blob Storage ²	Microsoft.Storage.BlobCreated
Microsoft.Storage.BlobDeleted	
Event Hubs	Microsoft.EventHub.CaptureFileCreated
Custom (Event Grid) Topic	Any custom application event.

The next updates to Azure Event Grid will extend support for the following Event Publishers: Azure Automation, Azure Active Directory, API Management, Logic Apps, IoT Hub, Service Bus, Data Lake Store, and Cosmos DB.

² The Blob Storage Event Publisher is currently in private preview.

Event Handlers

The event handlers that are currently supported are Azure Functions, Logic Apps, Azure Automation, and Webhooks. Future event handlers will include Service Bus, Event Hubs, Azure Data Factory, and Storage Queues. The webhooks Event Handler is versatile because any 3rd party service that supports webhooks can be used as a target.

Use case

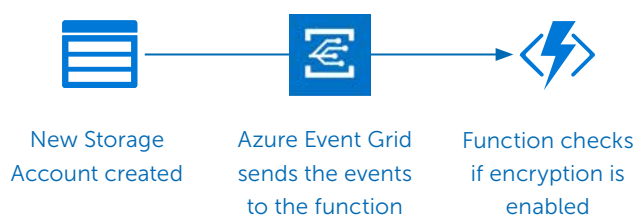
One of our customers is a banking company and made the strategic decision to move to Azure. The need for security compliance is very high, and one of the requirements for using Azure Storage Accounts is that they have to use encryption. We can build a script that will run through all the subscriptions, filter the resources of the type Microsoft.Storage, and then check whether they use encryption. This is not extremely difficult, but it is still a polling mechanism in which all storage accounts are checked again and again, even if they are not changed. Moreover, the script has to be scheduled and must run X times per day or hour, with the chance of lagging behind events.

Wouldn't it be better to be "notified" as soon as possible when something regarding a storage account has happened? This is where Azure Event Grid can help us!

This new service allows us to react as soon as something has happened and notify multiple subscribers (event handlers) of specific events.

Implementation

The following example shows how Azure Event Grid can fire up events to Azure Functions. The full source code can be found on Github³. There is also an ARM template available as Gist⁴ that can be used to create EventGrid subscriptions from a script or a deployment pipeline.



The first piece we need is a function that can parse the event data provided from the Event Grid. The schema is well documented and can be found at the following location: <https://docs.microsoft.com/en-us/azure/event-grid/event-schema#azure-subscriptions>

We will use the subscription events, which means that whenever something happens in a subscription (every time a resource is created, deleted, modified, etc.) an event will be published and picked up by Event Grid.

```

az eventgrid event-subscription create
[
  {
    "topic":"/subscriptions/{subscription-id}/resourceGroups/{resource-group}",
    "subject":"/subscriptions/{subscription-id}/resourceGroups/{resource-group}/providers/Microsoft.EventGrid/eventSubscriptions/LogicAppd584bdf-8347-49c9-b9a9-d1f980783501",
    "eventType":"Microsoft.Resources.ResourceWriteSuccess",
    "eventTime":"2017-08-16T03:54:38.2696833Z",
    "id":"25b3b0d0-d79b-44d5-9963-440d4e6a9bba",
    "data": {
      "authorization":{"azure_resource_manager_authorizations"},
      "claims":{"azure_resource_manager_claims"},
      "correlationId":"54ef1e39-6a82-44b3-abcl-bdeb6ce4d3c6",
      "httpRequest":"",
      "resourceProvider":"Microsoft.EventGrid",
      "resourceUri":"/subscriptions/{subscription-id}/resourceGroups/{resource-group}/providers/Microsoft.EventGrid/eventSubscriptions/LogicAppd584bdf-8347-49c9-b9a9-d1f980783501",
      "operationName":"Microsoft.EventGrid/eventSubscriptions/write",
      "status":"Succeeded",
      "subscriptionId":"{subscription-id}",
      "tenantId":"72f988bf-86f1-41af-91ab-2d7cd011db47"
    },
  }
]
  
```

The function that handles the event after parsing the event data will retrieve a reference to the storage account.

The next step consists of using the Fluent Azure Management libraries for .NET (<https://github.com/Azure/azure-sdk-for-net/tree/Fluent>) to check whether the Storage Account is using encryption:

```

AzureCredentials cred = new AzureCredentials(sp, tenantID,
    AzureEnvironment.AzureGlobalCloud);
IAzure azure = Azure.Authenticate(cred).withsubscription(subscriptionID);

string storageAccountID = eventData.subject;
string storageAccountname = subjectDictionary["storageAccounts"];

bool? isEncrypted = false;

//Get the storage account
var storageAccount = azure.StorageAccounts.GetId(storageAccountID);

//And now let's check if the Blob encryption is enabled
if (storageAccount.Encryption != null)
{
    isEncrypted = storageAccount.Encryption.Services.Blob.Enabled;
}

string returnText;

if (isEncrypted.GetValueOrDefault())
{
    returnText = $"Storage Account {storageAccountName} is Encrypted";
}
else
{
    returnText = $"Storage Account {storageAccountName} is NOT Encrypted"; ;
}

log.Info(returnText);
  
```

In this example we will output the result to the function logging console.

³ Source code : <http://xpir.it/xprt5-eventgrid>

⁴ ARM template: <http://xpir.it/xprt5-eventgrid-arm>

Creating the Azure Event Grid subscription

The easiest way to create an Azure Event Grid subscription programmatically is to use the Azure CLI 2.0. You can use this directly from the browser in the Azure Portal using the Azure Cloud Shell.

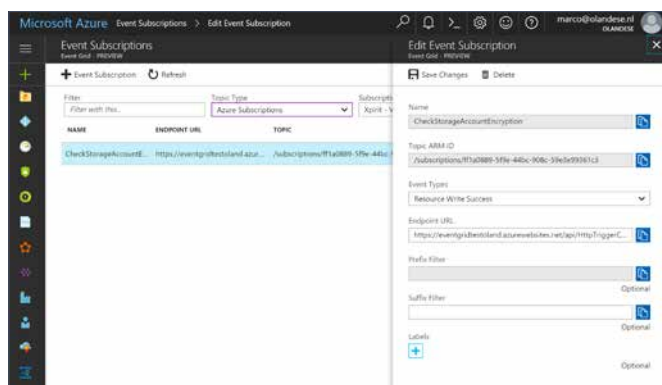
The following code snippet shows how:

```
az eventgrid event-subscription create
--name CheckStorageAccountEncryption
--included-event-types Microsoft.ResourceWriteSuccess
--endpoint "https://eventgridtest.azurewebsites.net/api/HttpTriggerCheckStorageEncryption"
```

As you can see, we give the subscription a name: "CheckStorageAccountEncryption", and we tell Event Grid to send the events to the URL (endpoint) of the function. We also add a filter, and because we are only interested in successfully created resources, we add "Microsoft.Resource.ResourceWriteSuccess".

Unfortunately, it is not possible to filter other event properties, e.g. the Resource Provider (Microsoft.Storage), so we have to do this after parsing the event in the function.

The "Event Grid Subscription" section of the portal shows the subscription that was created:

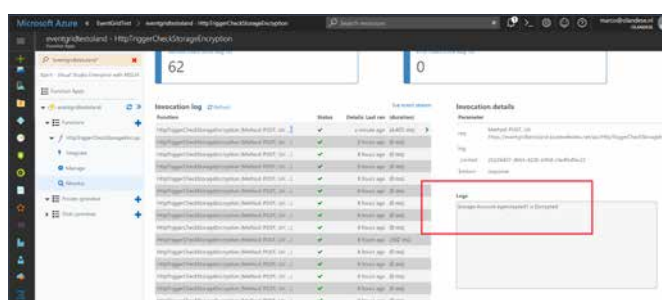


Working example

Let's create an encrypted storage account:

```
az storage account create --resource-group
EventGridTest --encryption blob --sku Standard_LRS
--name egencrypted7
```

The result is shown in the function's output log:



Conclusion

In an increasingly serverless world, Azure Event Grid is a valuable addition to create modern applications following an event-driven model. The integration with the Azure platform makes it an ideal candidate for Ops automation and micro-services scenarios.

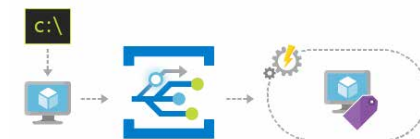


Figure 4: Ops Automation

However, it is certainly not limited to Ops automation, thanks to custom topics and webhook handler capability. These make it very suitable for many serverless integration scenarios, even with third party services.



Figure 5: Serverless applications

At the time of writing this article, the service is in preview.

In the future, Microsoft will add more event publishers and handlers to make the native integration in Azure and third parties even better, and to ensure that integration between different domains will become even easier and more reliable. Monitoring will also be added to allow inspection of ingress of events, event matches, pushes to event handlers, and dropped events.

Event Grid allows developers to focus on business processes when they "just happen", using fewer computing resources and without any worries about infrastructure. </>

"The Xpirit bearded Hipster who thrives on combining technology with photography."

Chris van Sluijsveld about Marc Duiker





From Search to Checkout without annoying your customers

How Accelerated Mobile Pages and Progressive Web Apps can boost online sales. In the world of e-commerce, customers are becoming increasingly mobile. In The Netherlands, 50% of consumers are shopping on their mobile phone. Among those under 35 years of age, mobile purchases are at 65%. Numbers for searching and browsing for a potential purchase is over 70% overall.

Author Gert Hengeveld

Converting these visitors into customers is a delicate task. Consumers are still hesitant to make mobile purchases. The challenge lies in optimizing the mobile user experience. Some shops have tried to improve user experience using a native mobile app. These apps are installed through the Apple or

Google app store. However, installation rates for mobile apps are decreasing. *The app boom is over.* Most smartphone users download zero apps per month. Unless you're a major player and consumers use your app on a weekly basis, chances are that your expensively built app is never installed.

So how can you get those smartphone users to make purchases? Luckily the web has evolved a lot in recent years. Features such as push notifications, real-time updates, seamless screen transitions and geolocation are all available on the web platform. Advances in front-end development

Alibaba.com increased mobile conversions by 76% with a Progressive Web App.

make it possible to achieve a highly interactive, high-performance user experience that matches a native app. The web also offers some advantages over native apps such as discoverability through search and social media, and not having to install anything. Key to a successful web app is user experience. Reduce friction in your sales funnel and more customers will make it to the end.



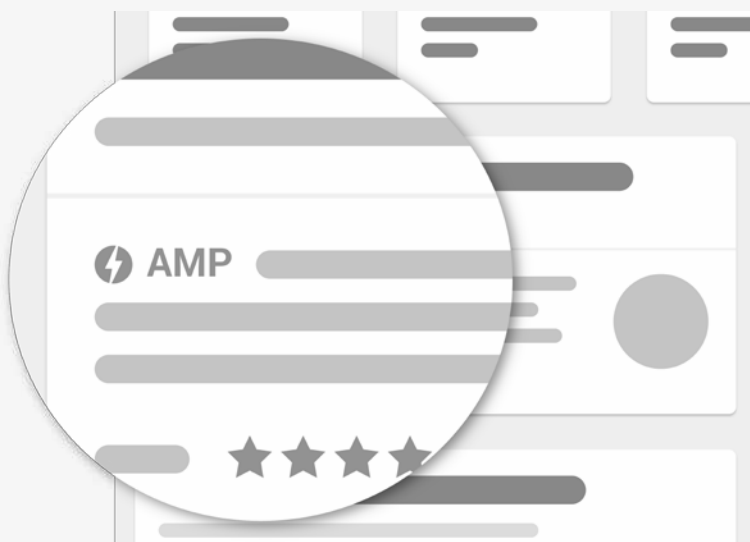
From Search to Checkout can be done without loading delays.

The mobile web experience

Two major breakthroughs in high-performance web experiences came from Google in 2015 as it introduced Accelerated Mobile Pages (AMP) and Progressive Web Apps (PWA). Two very different techniques trying to solve the same problem: how to make the web fast and reliable. Accelerated Mobile Pages is a way to build a lightweight version of your webpage that Google (and others) will make instantly available right from its search results, at least on mobile. AMP pages are very limited in functionality, but blazingly fast. They are also more likely to be at the top of the search results.

One e-commerce company recently deployed an immersive AMP experience and saw a 20–30% conversion uplift.

Progressive Web App is a broader concept aimed at describing a web app that offers certain features that make it fast, reliable and engaging. A PWA is a state-of-the-art web application that uses all of the power of the web platform, and does so responsibly. It's generally a single-page application, meaning page transitions are fast and smooth and we can offer a high-end user experience. A Progressive Web App is accessible from the web like any other website, but can also be "installed" by adding an icon for it on the home screen. Opening the app from the home screen will render it in full-screen mode, essentially mimicking a native mobile app. We can also implement push notifications so that customers can be actively engaged.



Google shows the AMP tag for search results that support it.

Accelerated Mobile Pages and a Progressive Web App together make for a very potent combination. Because an AMP page is instantly available from Google and AMP allows us to pre-cache our PWA in the background, we can achieve a sales funnel that goes from Google search to checkout without ever showing a loading indicator or staring at a blank screen. Combine it with a user-friendly PWA and you've got a sales funnel as smooth as silk.

```
<amp-img src="image.jpg" width="90" height="50" layout="responsive">
</amp-img>
```

The AMP project provides a predefined set of web components to build your pages in order to optimize performance across devices.

Driven by modern web technologies

The technology required to achieve this seamless shopping flow is about to become mainstream. Google introduced Accelerated Mobile Pages in 2015 but has only recently started encouraging it for e-commerce. Progressive Web Apps are made possible by new browser features such as Service Workers and the Push API.

It's a common misunderstanding that a Progressive Web App must be powered by a modern client-side UI rendering library. In fact, you can still render webpages on the server. All you really need in order to get a baseline PWA is an app manifest file and a service worker, which is just a tiny bit of JavaScript that can be bolted onto any webpage. However, this will not get anywhere close to offering that app-like feeling that we would like to achieve.


```

{
  "short_name": "Acme Shop",
  "name": "Acme Corporation Online Shop",
  "icons": [
    {
      "src": "launcher-icon-2x.png",
      "sizes": "96x96",
      "type": "image/png"
    },
    {
      "src": "launcher-icon-3x.png",
      "sizes": "144x144",
      "type": "image/png"
    },
    {
      "src": "launcher-icon-4x.png",
      "sizes": "192x192",
      "type": "image/png"
    }
  ],
  "theme_color": "#6c1d5f",
  "background_color": "#f8f7fc",
  "start_url": "/index.html",
  "display": "standalone",
  "orientation": "landscape"
}

```

A Web App Manifest makes your app look like a native one.

In practice, most companies that have deployed a PWA are indeed using a modern UI library such as React. These tools help us build high-performance user experiences, including fluent transitions and animations. As a bonus, we can usually run the same code to render HTML on the server. Server-Side Rendering makes sure that search engines and social media sites are able to crawl and access your content. Accelerated Mobile Pages must always be rendered on the server. This simplifies caching and avoids running heavy JavaScript code on a slow device. In effect this means that to combine a PWA with AMP, we have to render our pages both on the client-side and the server-side. To do this efficiently, it makes a lot of sense to run the same code on both sides using Node.js.

```

<!doctype html>
<html>
  <head>
    <meta charset="utf-8">
    <script async src="https://cdn.ampproject.org/v0.js"></script>
    <title>Acme Corporation Online Shop</title>
    <!-- ... -->
  </head>
  <body>
    <!-- ... -->
  </body>
</html>

```

The AMP HTML must be rendered server-side.

Building a stellar e-commerce experience

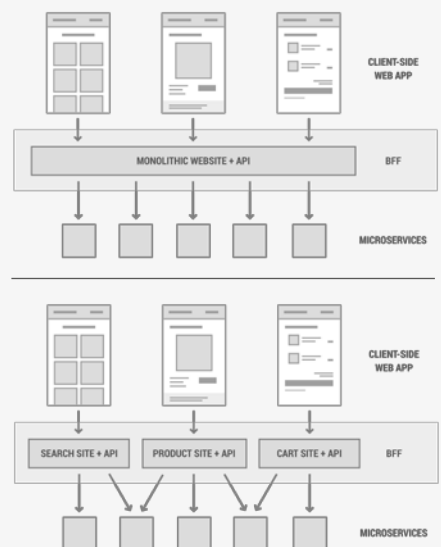
Following the previous paragraphs it should come as no surprise that the two primary ingredients for our e-commerce front-end are a PWA and AMP. Here's what it takes to build these and hook them up.

A high performance front-end framework with SSR capability

To get started, we need a tool to help us build the user interface and deal with client-side logic such as keeping state and handling transitions. In order to serve AMP documents without a lot of additional work, it will need to support Server-Side Rendering. Because a large part of our audience will be on mobile, performance is an important aspect. Luckily there are tools such as Lighthouse to measure performance and plenty of benchmark apps. Although React is the most popular choice in this space, Preact and Vue are very solid alternatives worth considering.

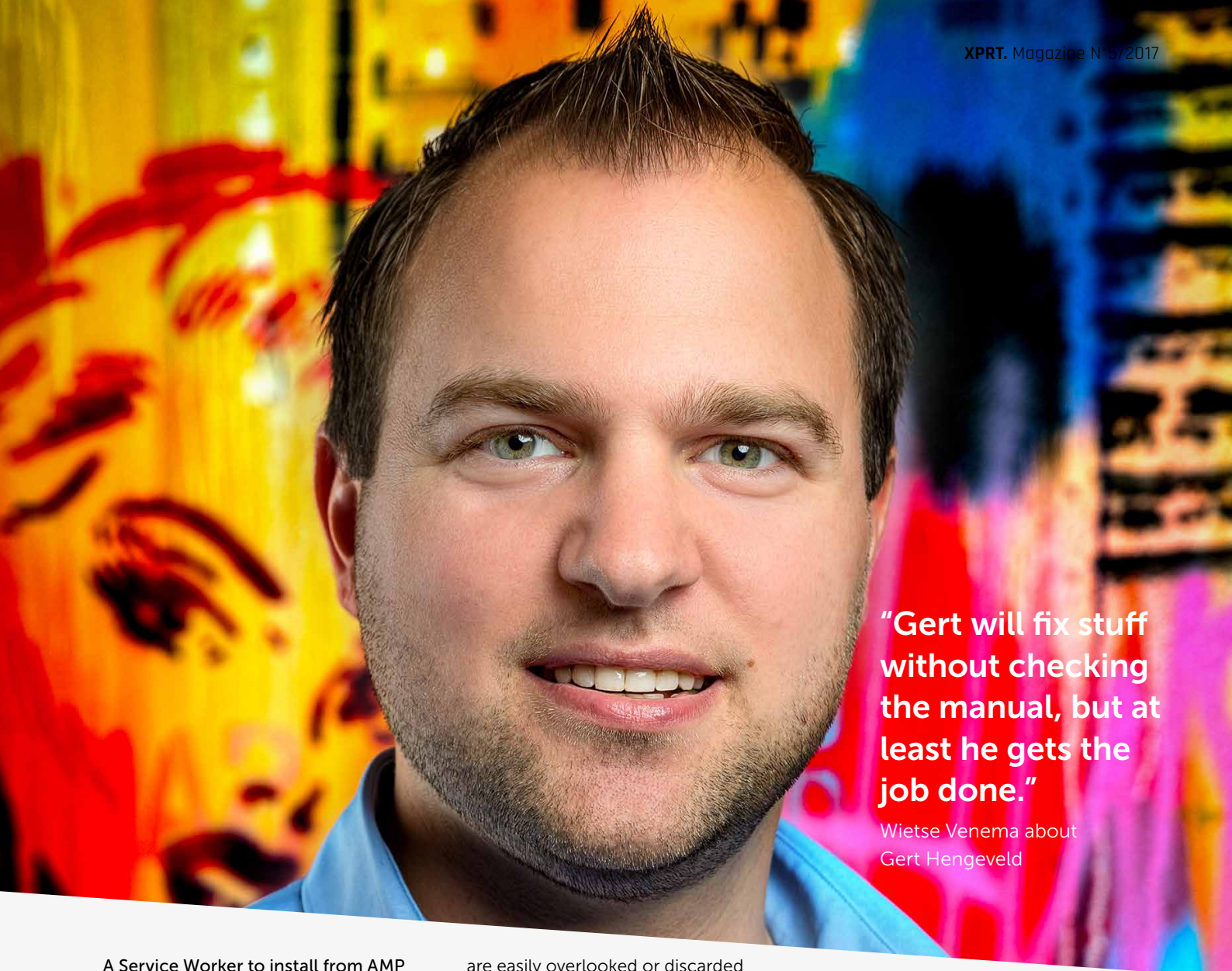
Setting up universal server-side rendering is still very complex. Luckily there are several frameworks which provide universal server-side rendering out of the box: Next.js for React projects and Nuxt.js for Vue. Both offer significant benefits over a plain React or Vue setup.

Back-end for Front-end (BFF pattern)



You may want to split up your back-end for front-end to create verticals and avoid a monolithic front-end.

AMP documents must be rendered on the server. We could of course do this in any back-end technology, but to be able to reuse interface components between AMP and our PWA, we should be using the same technology. As such we'll need Node.js to render our Preact components on the server. Running JavaScript on the server has the added benefit that your front-end developers can take ownership and responsibility for their entire product. In addition, many innovations in front-end optimizations are adopted by the Node.js community first, and generally very easy to apply. On top of Node.js, we'll need a server framework such as Express. We can still use our existing back-end services as backing for the Node.js service, so we won't be moving any important business logic.



“Gert will fix stuff without checking the manual, but at least he gets the job done.”

Wietse Venema about
Gert Hengeveld

A Service Worker to install from AMP

A Service Worker is what allows our app to work offline and enable push notifications. It's a script that can be installed in the browser and run in the background to cache resources and act on events. To make the transition from Google to our app appear instantly, we can use amp-install-serviceworker to install our Service Worker in the background while the customer is browsing our AMP page. This way we can pre-cache necessary resources for our app in the background so it will load instantly when the customer clicks through to the product detail page. Because the PDP is part of our single-page app, any subsequent page transitions can be instant too.

Attention to detail

Perhaps the most important aspect in building a high-quality web experience is attention to detail. There are many aspects to web development which

are easily overlooked or discarded because they are “too much work”. These include performance, user experience design and overall look & feel. Unfortunately, they are the first casualties of deadlines, budget restrictions and developer laziness. It's very sad to see a great concept being poorly executed, but in reality that's what happens. The key is to not settle for a sub-par user experience. It's better to build half a product with only a few well built features than to build a half-assed one with many poorly executed features. Setting a performance budget is a good way to keep an eye on performance. There's a whole list of things that we can do to improve it. Of course, performance alone isn't going to cut it. Consumers expect a polished product, especially on mobile. That means hiring UX designers and performing usability tests. Because we're targeting many

devices, taking a mobile-first approach to interface design and development is highly recommended.

Final thoughts

The web is evolving at a rapid pace. Google continues to push the web as a platform. Targeting the web as the primary platform makes a lot of sense for e-commerce, as smartphone users are unlikely to install a native app. New browser capabilities allow us to provide a high-end mobile user experience, while reaping the benefits of the web platform. Luckily, tools have evolved to make adoption of current best practices much easier than it used to be. Nowadays we can offer our customers an e-commerce experience that will not alienate or drive them away, but one that they will love. </>

Scaling Scrum to the limit

You're likely to have been asked the question: "we need to go faster, how many more people do we need?" Most people naturally understand that just adding a random number of people isn't likely to make us any faster in the short run. So how do you scale Scrum to the limit? And what are those limits?

Author Jesse Houwing



Meet Peter, he's a product owner of a new team starting on the greatest invention since sliced bread. It's going to be huge. It's going to be the best. Peter has started on this new product with a small team, six of his best friends and it has really taken off. In order to meet demands while adding new features, Peter needs to either get more value out of his teams and if that is no longer possible, add more team members.

He and his teams have worked a number of sprints to get better at Scrum, implemented Continuous Integration even to deliver to production multiple times per day. It is amazing what you can do with a dedicated team willing to improve.¹ But since their product was featured in the Google Play Store they've found themselves stretched to their limits. Peter has found himself in the classical situation in which many product owners and project managers find themselves. How do you replicate the capabilities of your existing team without destroying current high-performant teams? He contacts a good friend, Anna, who has dealt with this situation before and asks for her advice.

¹ <https://xpir.it/xprt5-scaling-scrum1>

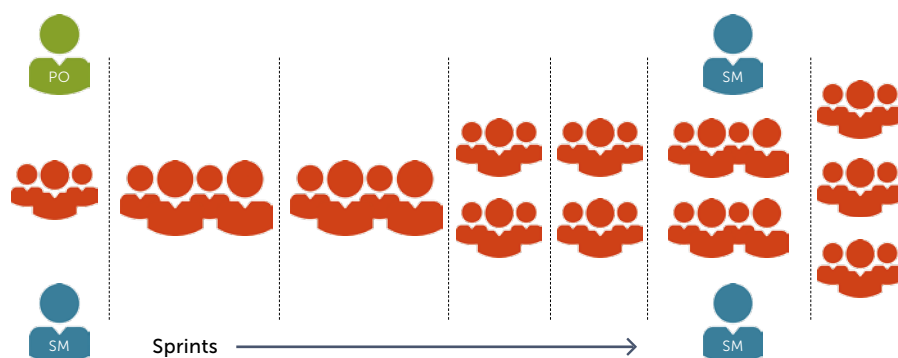


Figure 1: Grow and Split.

Anna explains that there are two options of gradual growth that have a very high chance of succeeding with limited risk to his productive team.

1. Grow and split model

In this model, new team members are added to the existing team, one team member at a time and taking enough time to let the new member settle before adding the next. Once the team reaches a critical point, a natural split is bound to happen and you'll end up with two or more smaller teams. Peter remains the sole Product Owner (a product is always owned by a single owner in Scrum), but as they grow they may add an additional scrum master to help facilitate and help the teams to keep improving.

Most often the split happens naturally when a team grows beyond a certain size. This allows the team to self-manage their new composition. However, the new teams may never perform as well as the original team.

2. The apprentice model

The second model, Anna explains, uses an age-old model to train new people on the job. In the apprentice model the existing team takes on two apprentices who are trained in the ways of working and the functional domain. After a couple of sprints these apprentices reach their journeyman status and start a team of their own.

The biggest advantage of this model is that the original team stays together. They do have to onboard and teach the apprentices, which is likely to impact the way they work together, but this model has a much higher chance of retaining the productivity of the original team.

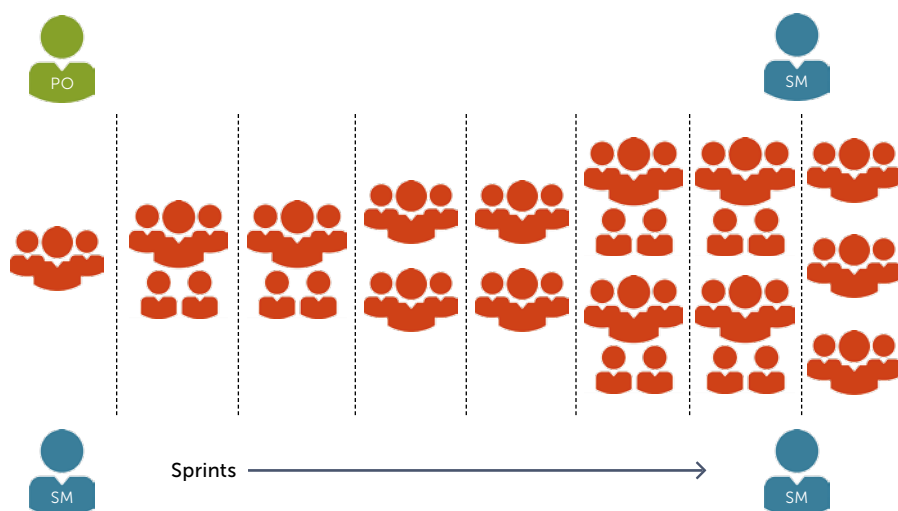


Figure 2: Apprentice Model

It may take a few sprints for the new team to reach the same level of productivity as the original team had, but you'll have a higher chance of keeping your first team stable and productive.

Unfortunately, in this model there are no guarantees either. Adding new people to an existing team can have lasting effects, even after these people leave. These effects can be both positive and negative. E.g. they may bring along a new testing technique that helps everyone become more productive, or they may involve new insights that cause division among the original team. Depending on the team, they may be able to benefit from both, but it may also tear them apart.

It's always the case that the original team will now have to learn to cooperate with the newly formed team, which will likely have a massive impact on their productivity.

Knowing when to stop scaling

Peter asks his team which model they feel most comfortable with and the team decides to start with the grow-and-split model, and after they've split off into two teams, adopt the apprentice model to grow further if needed.

He also asks Anna to join his company. She takes up the mantle of Scrum Master and focuses on helping the teams improve and helps them discover solutions for many of the problems introduced by working with multiple teams.

Meanwhile, Peter keeps asking the teams to train new team members and steadily the number of teams grows.

One afternoon Anna comes knocking on Peter's door and shows him a couple of statistics she has kept for as long as she has been working in the company. According to her statistics, she had been tracking the value delivered per sprint as well as each team's velocity - the latest additions haven't been able to really deliver more. She argues that the overhead of working together with so many teams has reached the maximum sustainable by the current architecture.

She asked the teams and found out that people are tripping over each other's work, integration regularly fails, and people are spending too much time in meetings and not enough time on "real work". Despite the practices she has introduced, such as cross team refinement and visualization of dependencies, it seems that they have reached the maximum size for the product.

While Peter is a bit disappointed, he has to admit that Anna warned him that he couldn't just keep adding people and expect an ever-increasing amount of work to be delivered.

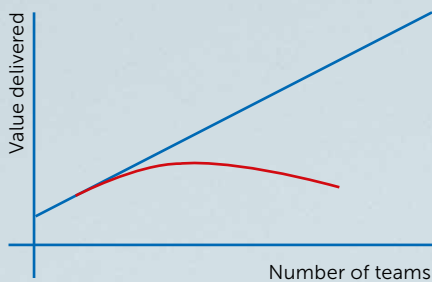


Figure 3: The sky may not be your first limit.

Useful metrics while scaling

While velocity (story points delivered), hours spent and number of tests passed are all viable ways of tracking progress for a development team, it's easy to measure the speed at which worthless junk is being delivered to production, without really knowing it.

This is why Anna also kept track of other metrics, such as value delivered, customer satisfaction (through app store reviews), incidents in production (through the monitoring tools they have in place) and more.² Keeping statistics about the amount of value delivered while you're scaling is important. You will probably find that while the total number of teams increases, each new team adds less and less value. This is a sort of glass ceiling that you may hit sooner or later. Breaking through it may require drastic changes to the application's architecture or to the way the teams work together.

As Peter and the original team never expected the product to take off this fast, the architecture of the application

was put together a bit haphazardly. And under the pressure to deliver, they cut a few corners left and right. He calls all of his teams into the company canteen and explains his predicament. Each team selects one or two of their most experienced team members and they form a temporary team of experts to figure out how to break up their little architectural monster. After peeling off a few functional areas and refactoring them into smaller, individually deployable parts of a cohesive functional unit, it quickly becomes apparent that this new architecture prevents them from tripping over each other's toes.

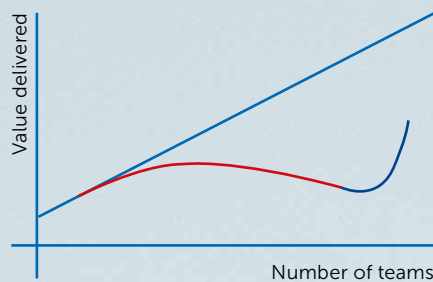


Figure 4: Drastic changes enable new growth.

You may have heard of this model before: small functional cohesive units of code that maintain their own data and that are called Microservices. These small units are ideal to form teams around and give these teams a lot of freedom.

Could we have done it differently?

Sometime later Peter finds Anna in the company coffee corner and asks her whether they could have taken another approach, one that would have shown the issues in their original architecture at an earlier stage of their product's development. He also wonders whether they could have scaled faster by hiring experienced teams.

Anna explains that there was a third option she never explained to Peter, because it carried a much higher risk, and she didn't dare risking the product. The third option was to quickly add a number of teams all at once, preferably teams of people who had already had some experience working together and that had experience working at such scale. At the same time, the original team members would be scattered

amongst the newly formed teams, optionally rotating to share their specific knowledge of the domain or processes, and to explain the architecture and infrastructure.

3. Scatter (and rotate) model

In this model, given that you hit the problems in the established processes and in the architecture head-on, everyone needs to work together to quickly find solutions to all of the problems they encounter. If they manage this, they may be able to quickly find a way to work together. They may, however, also completely come to a stand-still or the amount of conflict may reach levels unimagined before.

To ensure the new teams have equal access to the knowledge and skills of the original team, the original team members often rotate amongst the newly formed teams or they are not dedicated to any team for a few sprints, before everything settles down.

If this model had succeeded, they may have been able to scale much faster. However, they could also have been out of business.

Peter reflects that had they had a direct competitor in the market who was able to deliver much faster, they could have taken this risk. But it would have been an all-in gamble. He's glad they weren't in that situation.

Conclusion

There isn't really a hard limit in terms of how many people can contribute to an agile product or organization. But clearly there are limits to the pace at which you can grow, to the amount of control you can have over what is going on in every team, and to what the product's or organization's architecture and processes can sustain.

There are multiple models to grow your ability to deliver value. While adding teams may seem the easiest solution, investing in continuous integration, automated deployments, and a flexible architecture may deliver more sustainable value faster.

² <https://xpirt.it/xprt5-scaling-scrum2>

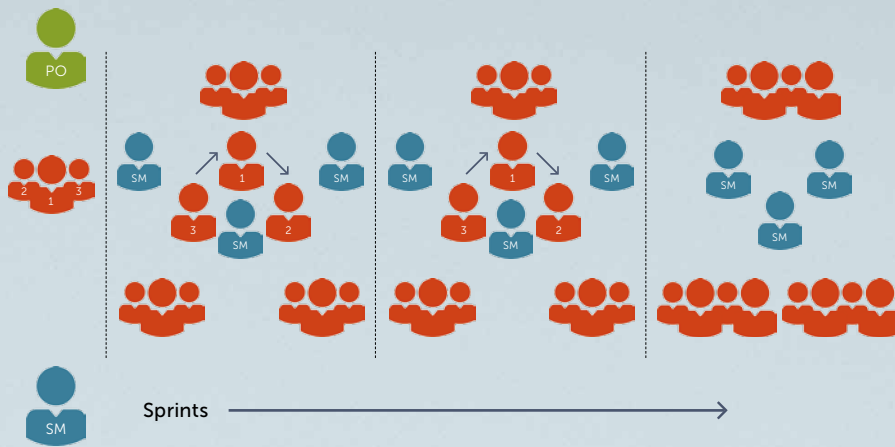


Figure 5: Scatter (and rotate) model

When you do need to scale beyond what's possible with a single team, remember that if you're not ready for it, you'll exponentially scale your team's dysfunctions. To be very blunt, if you scale shit, you end up with heaps of it. When you're able to deliver quickly, efficiently and professionally, you can scale your teams.

Keep measuring while you scale and keep evaluating your way of working, collaboration and architecture. Using your statistics, you can make an informed decision whether to scale further. Without them, you may be degrading your ability to deliver value without ever knowing it.

Keep inspecting your processes, tools, architecture and team composition regularly. Your team will probably know what to improve in order to deliver more of the right things more efficiently. </>

"Soon to bring the joys of Scrum to a place near you! Trainer, coffee connoisseur, ALM expert, Jesse is a great addition to any team."

Loek Duys about Jesse Houwing



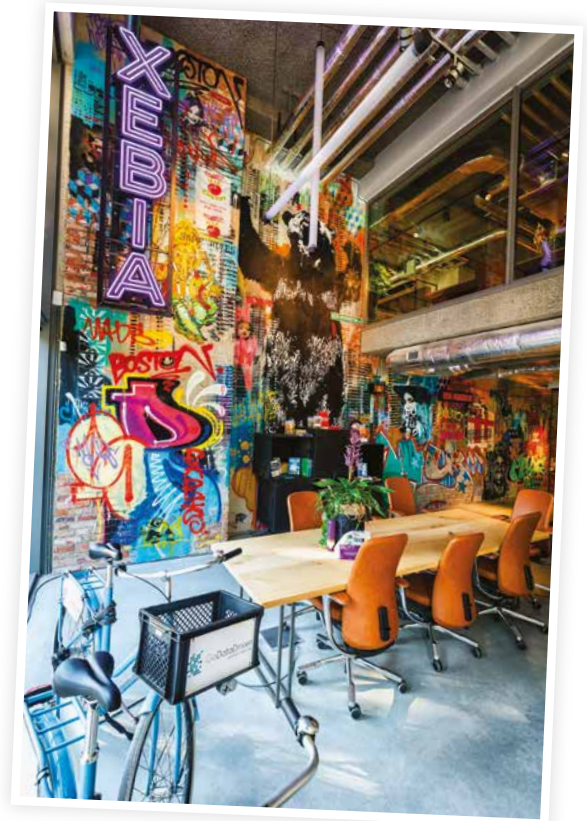
Empower yourself.

Do you have any training budget left? Use it now!

Get trained by the world's leading experts, scale your skills and learn everything you need to boost your career.

Register now by using discount code **XPIRIT10** and get **10% discount** on these courses:

Microservices with NServiceBus	7 & 8 November
Scaled Professional Scrum	20 & 21 November
Azure Beyond Cloud Services	23 & 24 November
Microservices with Azure Service Fabric	11 & 12 December



Visit [Xpirit.com/training](https://xpirit.com/training) or training.xebia.com for more details.



training.xebia.com

Xebia Nederland B.V. • Laapersveld 27 • 1213 VB Hilversum • +31 35 538 19 21 • training@xebia.com
Amsterdam Office • Wibautstraat 200 • 1091 GS Amsterdam

8 Years of CPU in a Day

Come to Valhalla with Azure DevOps #ftw

The idea seemed simple: bring the global community together so they could share and learn how to best use DevOps on the Microsoft stack using Visual Studio Team Services, Microsoft Azure, with Visual Studio 2017 and Xamarin tools. Do this one day a year and make the event global. This meant marshalling people, locations, and resources across the globe. But having a global meet up wasn't enough. Xpirit wanted it to be hands-on.

Authors Brian A. Randell & Ian Griffiths, DuoMyth

On June 16, 2017, Xpirit kicked off its first Global DevOps Bootcamp in New Zealand and finished at the Western part of the continental United States. With events in Australia, India, Europe, South and North America, we were chasing the sunrise to have VMs ready to go at each location. During the event, each attendee had access to their own private dual-core Windows virtual machine running in Azure. At peak, 1,500 VMs were running using 3,000 cores of compute and over 180 terabytes of storage. Each attendee had their own private environment to work with Visual Studio, Docker, and the other tooling needed to work with the hands-on labs and get their hands dirty with DevOps in a lab environment.

All made possible through hundreds of individuals putting hard work on the event, the power the Azure cloud, and Valhalla.

Come to Valhalla

Valhalla is the name of our solution built on top of Azure that makes events like the Global DevOps Bootcamp, the DockerCon 2017's hands-on lab pavilion, and even 20-person hands-on workshops easy to run providing one or more virtual machines for each attendee. The first event hosted on Valhalla was a Microsoft ALM lab back in early 2014. Since then, Valhalla has provided a solid foundation for running traditional instructor led training classes, road-shows for Microsoft in the United States, Europe, and Middle-East as well as DockerCon 2016 and DockerCon 2017 where each lab attendee received three Linux-based VMs and/or three Windows-based VMs work with various Docker and Windows Containers related labs.

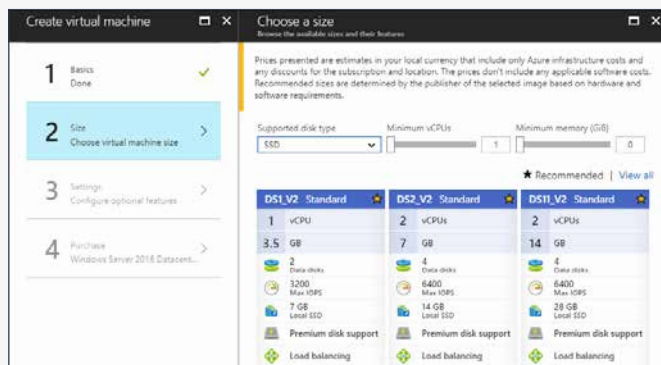
Valhalla was born out of a need to reduce the costs related to providing hands-on access to Microsoft's ALM stack. We had been working with folks in the Microsoft US subsidiary focused on helping them find ways to make on site events at customer locations and open events at Microsoft offices better by providing customers an opportunity to try the latest versions of Visual Studio and Team Foundation Server in a hands-on experience, on demand. We had been looking at using Azure hosted virtual machines. But there were some issues. One big non-technical impediment was the original billing model. In June 2013, Microsoft moved to a per minute billing model as well as they stopped charging for compute when VMs were not running. This change pushed things over the proverbial hump. We knew we could build an affordable system on top of

Azure. Moving from just ALM related content, it expanded to be a general-purpose system to provide “students” with access to a one or more VMs, all within a managed environment in the cloud.

Over the summer of 2013, we designed the system and figured out funding. In September, we started writing Valhalla originally in partnership with our friends at Endjin. In the beginning, the VMs were to run on top of the IaaS infrastructure where each virtual machine was allocated with a Cloud Service. We designed the system to support a single payee model as well as a shared payee model. We used web sites hosted in cloud services, pushing messages to queues and worker roles to process commands from the queues with table and blob storage as our persistence stores. We wanted to make the system easy to use and manage yet be flexible when it came to the types of content to support. In the beginning, we only supported Windows VMs. As Azure changed, so did we, adding support for Premium SSD storage and Linux VMs.

Valhalla Today

Naturally, creating a single virtual machine in Azure using the Azure Portal is easy. The UI guides you through the steps until you click the final button.

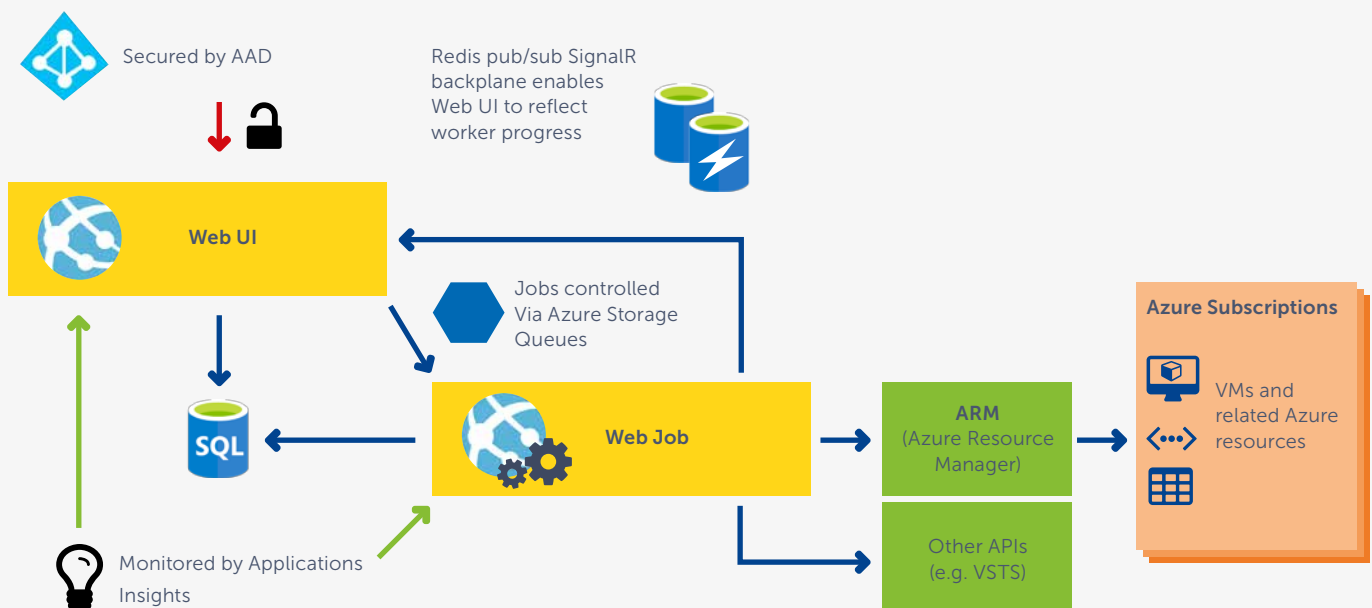


If you’re running regular events or even one large event you’ll obviously want to automate this—nobody wants to walk through this VM creation wizard 1,500 times. And while you might think just a little bit of PowerShell or an ARM template makes it all “simple”, it turns out it’s a bit more complicated. From the beginning, we designed Valhalla to support flexible class deployments (type of VMs, number of VMs, number of delegates) as well as multiple subscriptions. In fact, having multiple subscriptions is a key way to scale with Azure both for performance as well as scale out. If you run events needing many hundreds of VMs you will rapidly discover that the simple resource structure you get if you create a VM through the Azure portal does not scale—you will run into Azure’s per-subscription resource limits, such as the default limit of 50 Virtual Networks. You can get this limit lifted with a support request but there’s a hard limit of 500, so you can’t use the simplistic one-VNET-per-VM model beyond that point (but equally, there can be issues if you put all your VMs on a single VNET). With careful resource design you can create thousands of VMs in a single subscription but you are then likely to run into Azure’s per-subscription API rate limiter, which can slow you down or even cause operations to fail entirely. So multiple subscription support becomes a must-have at sufficiently large scale.

At the simplest level, we want to provide a person with access (RDP or SSH) to one or more virtual machines for a period of time. In most scenarios, we provide access to the delegates via a custom e-mail message that we send out using the SendGrid service. That said, proctors at an event can hand out the delegates access information instead.

We define what VM(s) a user gets in something we call a *bundle definition*. A bundle defines a number of pieces of data including the recommended Azure VM size to use, whether the VM needs to be accessible through a public IP, and more.

Valhalla Architecture



For storage, Azure supports two types for virtual machine hard drives (VHDs): basic and premium. The difference at its core is simple: premium storage is backed by solid-state drives (SSDs). While more expensive to run long term, we've built optimizations into Valhalla to keep them around only as long as needed. You only pay for the time the VHDs are allocated. Premium storage can provide a more performant experience when using interactive Windows sessions in particular. We support just about any VM that you would have access to in an Azure subscription.

Valhalla Architecture

While Azure Storage is used for VMs, we use SQL Database (Microsoft's name for its cloud version of SQL Server) as our main persistence store for tracking classes, resources, etc. In earlier versions of Valhalla, we used Azure Table Storage. While providing good performance and cheap storage, the programming model left a bit to be desired. We found the more traditional data programming model in SQL Database more productive. We use SQL Database to store most data about the system.

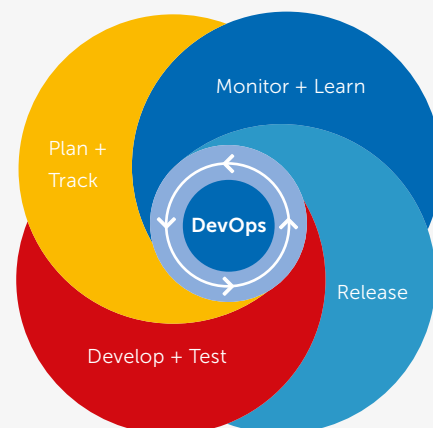
Again, as we've evolved Valhalla, we changed how we handle security. Our original system had its own role-based security model. Our current version relies on Azure Active Directory. Currently our system is mainly accessed by ourselves and customers who need to manage classes and events. Delegates don't need to log in to use their VMs. However, Azure AD's flexibility means we can support other Azure accounts, MSAs, Google, Facebook, and more.

As a cloud-based solution, the main UI of the system is implemented as an Azure App Service Web App. We use slots to make it easy to deploy new versions of a site.

Web Jobs handle long-running requests. Any time Valhalla needs to do something in Azure, whether it's scanning an Azure subscription's storage accounts for newly-updated VHD images, or creating VMs for an event through the ARM (Azure Resource Manager) API, that work runs in a Web Job.

We use Redis Cache to enable the Web Job to provide progress notifications for its long running work to end users. We are using SignalR to enable our web servers to push notifications to browsers, and we are using SignalR's Redis Cache backplane to make it possible for our Web Job to generate notifications that will be routed to whichever web server is managing the connection back to the relevant end user. Under the covers, this uses Redis Cache's pub/sub mechanisms.

We use Application Insights mainly for diagnostic purposes. If something goes wrong, Application Insights is very good at providing a holistic view—you can track an operation's progress through from some end user's browser through to the Web App and then on through the Web Job. Application Insights provides automatic interception of any operations that use Azure Storage or SQL Azure, and the ability to discover all of the events relevant to a particular request make it easy to get a good overview of everything that happened up to the point of failure.

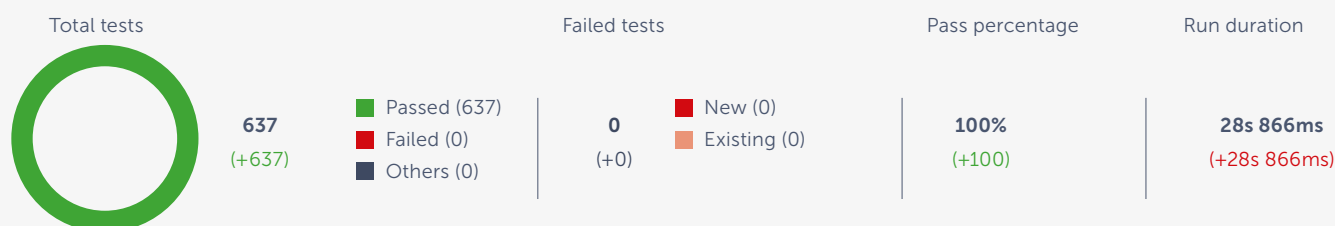


DevOps

At this point it's pretty clear that Azure provides us with flexible platform for our system and power on demand for our customers. However, with just two guys running the company, it helps to have a good DevOps process. Donovan Brown from Microsoft likes to say that "DevOps is the union of people, process, and product to enable continuous delivery of value for our end users." One advantage we have is we've been doing software development for over fifty years between us. And our general mindset is to change our process as needed to do more with less so we can deliver value. Thus, we've got the people and process part down pretty good (knowing it's a journey not a destination).

From a product perspective, we're all-in with the cloud with Visual Studio Team Services (VSTS). Back in 2013, it wasn't completely obvious but we started managing our source code using Git repos in VSTS. Over time, the flexible branching model has made it easy for us to work distributed

Test Results





Ian Griffiths

Ian is an independent consultant, developer, speaker, and author. He has written books on Windows Presentation Foundation, Windows Forms, and Visual Studio. He lives in London but can often be found on various developer mailing lists and newsgroups, where a popular sport is to see who can get him to write the longest email in reply to the shortest possible question.



Brian A. Randell

Brian A. Randell is a Partner with MCW Technologies LLC. For more than 20 years he has been building software solutions. He educates teams on Microsoft technologies via writing and training—both in-person and on demand. He's also a consultant for companies small and large, worldwide, including Fortune 100 companies like Microsoft. Brian lives in Upland, CA. When not working, Brian enjoys spending time with his wife and two children who enjoy making him look bad on the Xbox One (with and without Kinect). He loves watching movies, listening to heavy metal, and driving his 2008 E60 M5.

across two continents and eight hours' time difference. We use the Scrum template with Product Backlog Items, Tasks, and Bugs to plan and track our work. Our sprints are generally thirty days.

Naturally we care about quality and thus we've worked to build various automated quality checks into our process. It starts with unit tests that we run from within Visual Studio as well as during our automated build process. Our build process produces deployable packages, and runs a full suite of unit tests and integration tests. Our builds run using a continuous integration off master, and any feature branches which is nice when we're working on different features.

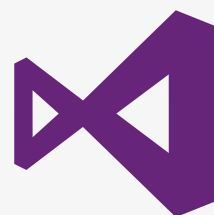
Our release process has evolved over time and this is a critical area to keep us moving forward. With our various customers and events, updating our bits in Azure is not something we want to be doing manually. We were early adopters of the "new version" of Release Management that's built into VSTS and it has served us well. We don't do a typical pipe line development from dev to test to prod. What usually happens is we do lots of deployments to one of our dev environments, then to test environments followed by prod. The key is that we can have multiples of each type of environment. They're different by purpose and customer but not by code. In addition to the deployment task, which involves pushing out any core service changes via ARM, the web sites, web jobs, as well as handling slot swaps, we use Selenium to run automated regression tests against the user interface of our web site. We have a specialized release definition for running these end-to-end tests that creates a whole new Valhalla environment (with the full set of Web Apps, SQL Server etc. deployed to a dedicated resource group) so we can test all functionality from scratch on a newly-deployed system, and this release then tears down the environment if the tests all complete successfully.

As a side note, we use Azure Key Vault to manage secrets such as SQL Server passwords and SendGrid credentials as part of DevOps. This enables us to avoid storing any secrets in source control, and also to keep them out of our build configuration. (VSTS's ARM deployment task is able to look up the secrets itself when it needs them by accessing the Key Vault directly.)

We track a wide range of data per event such as users, VM allocations, etc. in addition to data gathered from Application Insights. We also work to have post-mortems with customers to improve the system. New feature work comes from feedback as well as from our experience at running events, large and small. The larger events, like the Global DevOps Bootcamp, help us add make the system more robust for all events.

Lessons Learned

The saying goes "there's no place like production". Over the years we've learned a number of things running on Azure. One key learning is that multiple Azure subscriptions are necessary for large scale due to hard resource limits and ARM API rate limiting. Another key learning is test, test, test. Repeatability is vital. End-to-end testing is especially valuable—our full-system tests have caught more regressions than anything else. And as always, the only constant is change. Azure is dynamic and is a fantastic platform on which to build solutions. It's amazing to think that we can use over 180 terabytes of storage, have 1,500 VMs spun up burning eight years of CPU in a day and then give it back to the cloud. Come to the cloud, we think you'll like it. </>



Cloud Transitions done right!

It is no longer a question whether your organization will move applications to the cloud; it's only a matter of when and how it must be done. In this article, we will share our insights on what is required to make this transition successful. We will highlight various perspectives that should be taken into account when you consider a cloud migration and explain how you can determine the right strategy to follow.

Author Marcel de Vries & Martijn van der Sijde

Do you have a business case for migrating to the cloud?

When we talk with organizations about cloud transitions, we see that a lot of different approaches are taken.

The reason for this is that the chosen approach and steps to take are highly dependent on the perspective of who in the organization is asked to lead the transition.

If you ask the developers how they can move their application to the cloud, they will come up with great plans on how to change their application's architecture to micro services and how they can use the latest .NET Core framework, since it has been optimized for cloud workloads. This will lead to a high investment before things can be moved, because the fundamental differences of this type of architecture require the application to be rewritten.

If you ask the IT operations department how to make the transition, they will come up with a new way to provision infrastructure and set up a service catalog from which customers can request new virtual machines that will now be provisioned in the cloud. Furthermore, you will see extensive

network architectures and a lot of complexity, because they try to implement their current systems using cloud infrastructure, which is quite different when you want to get the maximum benefits from the cloud.

These are two examples of many other perspectives. Are they wrong? We don't think so. But we do think these approaches are suboptimal and will incur high costs and low return on investment. To prevent this, an answer should be given to the question: which migration strategy will contribute to your organization's business and IT goals? In other words: what is the business case for migrating an application to the cloud?

From CAPEX to OPEX

The cloud is a real game changer. Not only from a technical perspective but even more so from an economical perspective. In the past, an organization had to spend significant amounts of money to start a competitive online service. However, these capital expenses (CAPEX) are mostly gone, and all costs are moving to operational expenses (OPEX). This is because you don't have to invest in hardware, but instead you pay the cloud provider

for the resources you use. This is the on-demand, Pay-As-You-Go nature of the cloud. From this shift, we can see two forces that require our customers to change the way software is delivered. The first force concerns independent Software Vendors (ISV's) that are now asked to provide their Software-as-a-Service, because their customers want the same model for the software they buy as they now do with hardware in the cloud. The second force concerns enterprises that are driven to reduce their operational costs and one way to make this happen is by adopting the cloud. You see many enterprises state in their plans to totally move to the cloud and get rid of their own datacenters. This sounds very lucrative at first, but sometimes one tends to forget that just moving your existing machines to machines in the cloud is not at all economically beneficial. Your overall costs will probably become much higher.

CAPEX = Capital Expenditures, investment costs for developing a system

OPEX = Operational Expenditures, the returning costs when using a system

How do you move to the cloud the right way?

The first thing to understand is that moving to the cloud is not a matter of one size fits all. For example, if you are the ISV as mentioned in the previous paragraph, you need to look at your current software and determine the cost involved if you are now hosting this software yourself. You are now confronted with the incurred costs your customers had. These costs should be replicated for every customer you have. You need to look at what is the state of the software and in what part of the lifecycle it is. Has it just been built, has it been out there for a long time and does it already need significant rework, or is it a product that is at the end of its lifecycle and you need a way to provide SaaS but you don't want to invest?

Depending on the lifecycle phase of your application, you can project it on

a cloud migration strategy model such as the Gartner 5-R model (Rehost, Refactor, Revise, Rebuild, Replace), the Azure 5-R model or the AWS 6-R model. These "R" models state that you need to pick one of these strategies based on your company's cloud migration goals as well as the requirements and constraints of the specific application.

To demystify the options you can choose from, the strategies from the various "R" models are combined and explained in the following table.

After you have selected one of the strategies, you need to look at two factors. The first factor is the Capital Expense if you choose to Rehost, Refactor, Rebuild or Replace. With all these strategies, you need to invest in your solution before you can run in the cloud. Next you need to look at the

operational expense of running this application in the cloud for the next 5 years. This will result in a graph that can show you the total amount you will spend in the next 5 years, given a selected strategy.

To give you an idea of the result of this approach we will give an example of one of the cases we have seen in the field. The following graph depicts a 5-year plot of the capital and operational expenses of a product that needed to be moved to the cloud.

The problem statement that needed to be answered in this case was: how can we move our product to the cloud as quickly as possible with a capital expense as low as possible and still have a cost-effective operational expense for running the product for our customers in the long term.

Strategy	Description	Pros & Cons
Remove/Retire	Turn off applications from the portfolio that are no longer useful/not contributing to business goals.	+ > Positive impact on business case > Frees up time to spend on other applications
Retain/Revisit	Applications that have no priority for moving now. Leave as-is and evaluate again when most of the application portfolio has been shifted to the cloud.	+ > Focus on applications that deliver most business value
Rehost	Lift-and-shift your solution from on-premise to IaaS. Practically this means moving your current (virtual) machines to virtual machines in the cloud. No changes are made to your code. Another way of rehosting is containerization of your application.	+ > No changes to the code (minimal changes in case of containerization) - > No utilization of cloud benefits like scalability
Refactor/Replatform/Revise	Move to PaaS with minor adjustments, making relatively small changes to the existing application so it becomes more efficient in resource use, especially when you are serving multiple customers with the same software. This is better known as making an application multi-tenant.	+ > Better OPEX cost model compared to Rehost > Reuse of code considered as strategic or differentiating - > Cloud lock-in on PaaS
Rebuild/Refactor/Re-architect	Cloud-native move to PaaS, completely replacing the application and maximum use of cloud benefits. This is the developer's dream, where you go in the whole way, and use as much as possible Platform-as-a-Service from the cloud, so you obtain the best operational expense model.	+ > Full cloud-native benefits such as scalability > Most optimal OPEX cost model for custom software - > Lock-in when using PaaS services
Replace/Repurchase	Discard the application and move to a subscription-based Software-as-a-Service product.	+ > Fully outsourced application - > Possible data lock-in > Potentially difficult to customize and integrate

Table 1: Cloud migration strategies

Transition Strategy

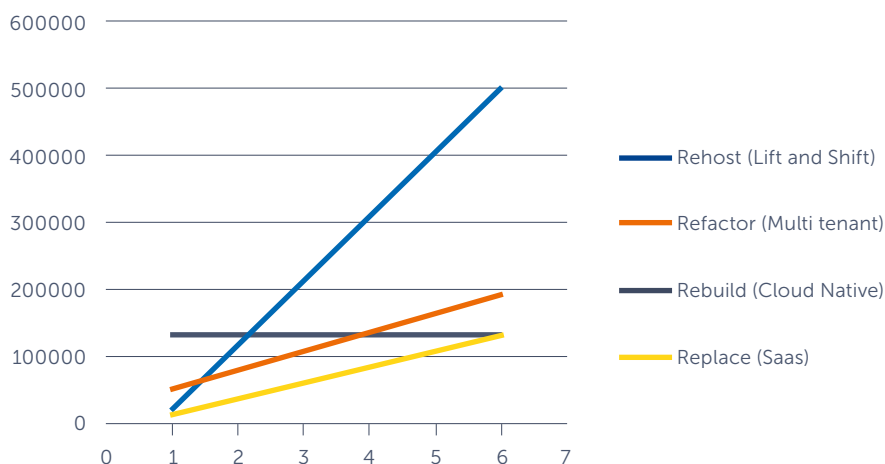


Figure 1: Cost comparison

For this specific case an estimate was made of the number of users of the product in each year and of the required changes to the product for each strategy.

When you look at this graph you see that a Rehost strategy will have a low upfront investment (capex) in the case of this product, but operational expenses make the total costs increase linearly. This is due to the costs of required virtual machines being used each year.

The Refactor strategy focuses on changing the product to make it multi-tenant, which is more resource-effective than a Rehost strategy. This is because the product will no longer require dedicated resources for each customer.

The Rebuild strategy will have the highest initial cost, because of the impact of required changes to the product. On the other hand, annual operational costs are relatively low because of the optimal usage of native cloud services, resulting in zero server maintenance.

The Replace strategy will substitute the product for a Software-as-a-Service solution. This is an interesting scenario from a cost perspective. The subscription of the product will be paid each period. The important question here is whether the product is a strategic differentiator. In other words: does the product contribute to innovation and differentiation of the product compared to the competition.

Looking at the problem statement, the question is: when does it become economically viable to make the investment to change the product, allowing the investment to be earned back in the long term.

You can see that the Rebuild strategy has the best operational cost model. After the initial investment, the cost line hardly increases compared to the other cost lines. Eventually it will be the cheapest, but the return on investment will take a couple of years because of the high initial costs. Another drawback is that time-to-market of the product will be longer compared to strategies requiring less significant changes.

If you only do the lift-and-shift, you will see that within one year it will be more expensive than refactoring the application to support multi-tenancy. The latter scenario is cheaper because one instance of the application will provide service to multiple customers whereas a dedicated application provides service to a single customer.

Based on this graph you could deduce a strategy of starting with a lift-and-shift, then start the investment first for multi-tenancy (Refactor) and then Rebuild it to cloud-native. Since it probably won't make a lot of sense to refactor and rebuild at the same time, you may want to choose a scenario where you look for options to gradually refactor towards a full cloud-native product.



“Thinks about architectures so vast and huge, that his 2 meter arm span is not wide enough to illustrate the range and size of things he talks about. ”

Jasper Gilhuis about Marcel de Vries

“Even though I could recognize Martijn for his analytical skills, I rather focus on his friendly appearance and ability to connect people.”

Alex de Groot about Martijn van der Sijde



The main benefit of this example is that this way of assessing applications will yield a set of possible scenarios and it will give you insight into a hard business case to make the investment and determine the best strategy.

An Enterprise perspective

When you are in an enterprise organization you can also make this same kind of assessment, but the difference will be that you need to deal with a portfolio of applications. In this case, you are best off to first divide the applications roughly into three categories.

- > Custom-built
- > Custom-built by a partner
- > Off-the-shelf and hosted on-premise

The category Custom-built involves software that is built to make a difference for the company. Gartner calls these systems "Systems of Innovation". These systems can be assessed in exactly the way we described for the ISV and from that you can pick the best scenarios.

The category Custom-built by a partner includes the systems of differentiation, where you often buy a partial off-the-shelf solution, but fully tailor this to the needs of the organization. These systems can also be moved to the cloud. In this case, you often ask the partner that built the product to take care of this.

The category off-the-shelf, usually means "Systems of Record". These are the systems you need, but everyone has the same and it is just there to ensure that you can run your business. There is no way you can gain an advantage by doing this differently from your competitor. For this category, you can often just move to the Software-as-a-Service solution of the vendor.

How to assess the applications you will move?

When assessing the applications that are going to move to the cloud, they need to be assessed on multiple levels. As stated before, it is important to determine the current phase in the application lifecycle. Next: where do we want to go with this product? At one time it may have been a "system of innovation", but by now each competitor also has it. This is the moment you choose to replace it with a SaaS solution. If the system will still be differentiating or innovating, it makes sense to look at the scenarios Rehost, Refactor and Rebuild. Practically this means that the initiative for cloud transformation allows you to Revisit the strategic importance of applications in the portfolio, and gradually migrate towards the cloud. The following figure shows the flow of assessment.

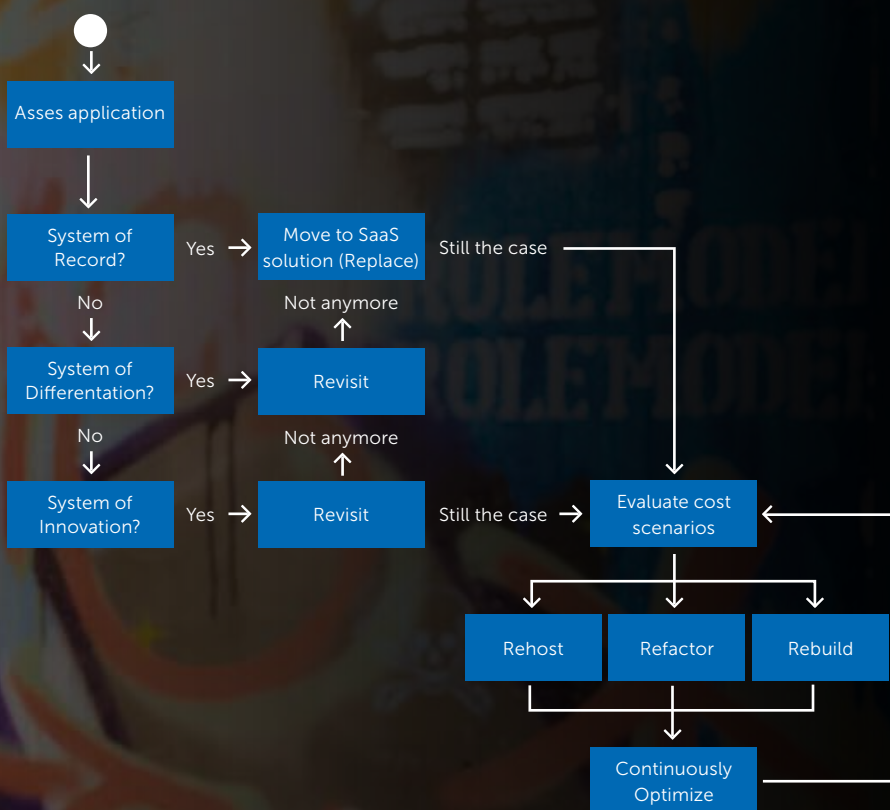


Figure 2: Decision tree for application portfolio assessment

Cloud transitions done right

Looking at cloud migration strategies there is no single strategy that will be appropriate for all applications in the portfolio of an ISV or enterprise. A mix of different approaches is required, based on the value that an application delivers versus the costs (investment and operational) of any selected strategy. Because these strategies depend highly on the situation, application, and types of cost involved, there is no one-size-fits-all solution.

In this article, we have described an approach for creating a business case for your applications when you move them to the cloud. If you ask your development team, you will get a different solution than when you ask your operations team. The major difference is that we have added the economical perspective, and this will allow you to create a balanced view on how to make the transition and predict the costs and benefits. </>

Containerized Testing

Many organizations nowadays are implementing continuous delivery practices to accelerate their time to market. An important part of continuous delivery is automated testing. However, a lot of companies are still struggling with how to do this in an effective way. Although a lot of test automation practices have appeared over time, the effort and time required for testing is still a bottleneck for many organizations. In this article you'll learn how the power of containerization can be leveraged to shorten your feedback cycles, reduce testing effort, and accelerate your time to market.

Author Cornell Knulst & Sander Aernouts

How automated testing is done nowadays

Containerized testing is a new practice within the space of automated testing. Before we look at what containerized testing has to offer in addition to other test automation practices, let's briefly look at commonly used practices nowadays.

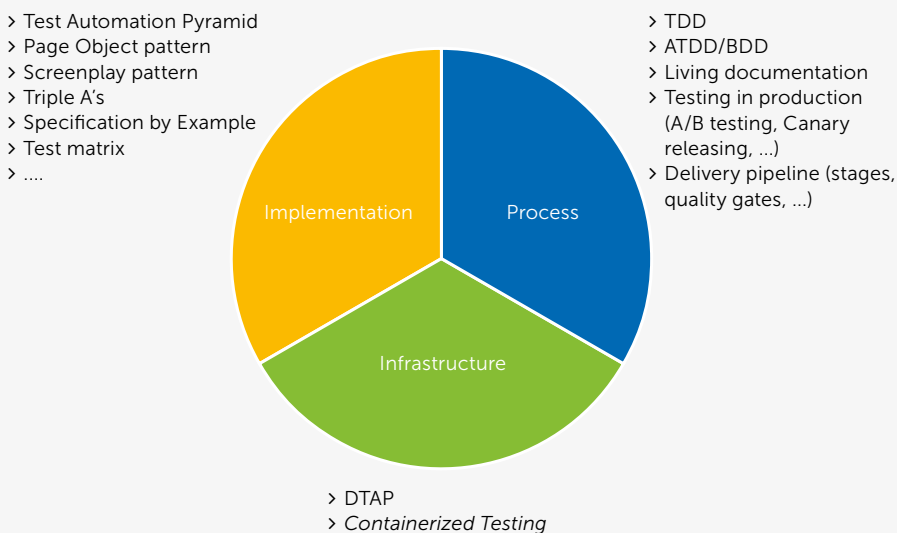


Figure 1 - Common test automation practices

A lot of current practices have to do with the implementation of your tests. One well-known practice in the domain of test automation is the Test Automation Pyramid¹. This pyramid describes that you implement your automated tests at the lowest pyramid level possible. The lower the level in the pyramid, the more autonomous, granular and maintainable your test cases are, and the faster your test execution. Other successful practices related to test implementation include the Page Object Pattern and Screenplay Pattern for creating maintainable UI tests.

In addition to implementation-related practices, process-related practices also appeared to be very suitable for automated testing. For instance, Acceptance Test Driven Development (ATDD) enforces development teams to create automated acceptance tests before actually coding the functionality. To deal with tests that can't be automated, as the Test Matrix² tells us, the practice of "Testing in production" was introduced to shorten time to market by executing those tests in production. For example, instead of executing Usability Tests as part of the delivery process, A/B testing can be used to execute this type of test in production.

Thanks to the above-mentioned practices, a lot of organizations are able to reduce the time required to deliver new functionalities into production. But if we look at the time, effort and costs that are needed to set up and manage a testing infrastructure, it is remarkable to see that only a few practices exist in this area, for example DTAP. Looking at the operational costs and realizing that the testing infrastructure is idle most of the time, wouldn't it be better to have the testing infrastructure on demand?

¹ <https://xpir.it/xprt5-container-testing1>

² <https://xpir.it/xprt5-container-testing2>

Luckily, we found a solution that can help us fill this gap. We call it containerized testing.

The shift to containerized testing

Many of you may already have discovered the power of containerization for your applications. Think of benefits such as scalability, freedom in hosting, immutable images, etc. So why not leverage the same benefits for our testing infrastructure. Wouldn't it be much easier to just set up your test infrastructure only when you need it?

What is containerized testing?

If we look at how a testing infrastructure is set up today, you will most likely have one or more dedicated, pre-provisioned test environments to support the execution of different types of automated tests.

Typically each test environment can only run specific types of tests. For example, performance tests are often executed on an acceptance test environment. Test or acceptance environments are often also shared across teams, which means that you will have to wait for that environment to become available. An example environment for running integration tests could look like the environment shown in figure 2.

Test environment

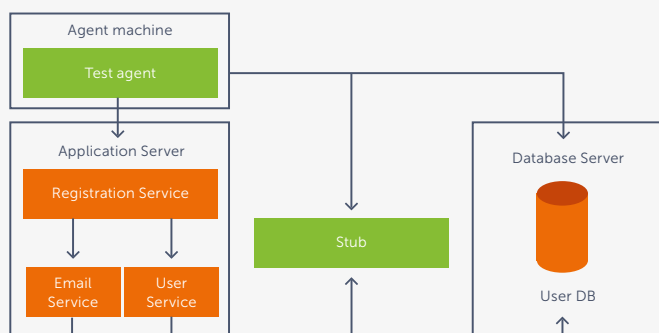


Figure 2 – Traditional setup of test infrastructure for a test environment

Figure 2 shows a fictitious system that is being tested. This system consists of a registration service, an email service, and a user service. The test agent is responsible for executing the automated tests, replacing an external dependency with a stub, and preloading a database used by the user service. Depending on the type of test, the database could be preloaded with test data to support different scenarios.

With containerized testing we do the same as we do with containerization of our applications, i.e. the environment becomes part of the test deployment. For every test we set up the required containers and configure the required environment. This is also known as configuration-as-code and infrastructure-as-code. By doing this, test environments are no longer a physical thing, but they become blueprints containing the various containerized components that we need to put together to execute a specific type of test. This concept of blueprints is referred to as environment-as-code.

For example, you will have different environment blueprints for security, performance, integration, and end-to-end testing. The nice thing about this approach is that we can set up a given environment on-demand, depending on the type of test we want to execute. Figure 3 shows an example of a containerized test environment, the same environment as we had in figure 2. However, it uses containers for the application under test and all other test infrastructure required to run the tests.

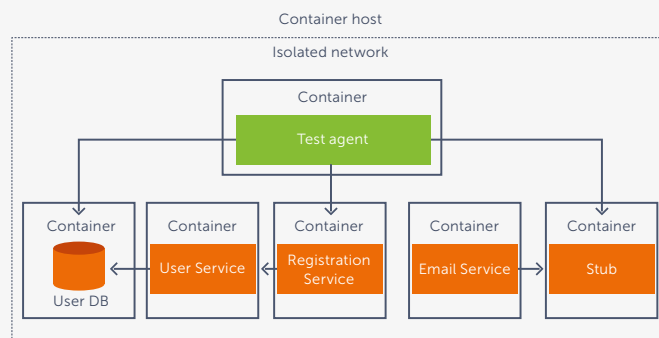


Figure 3 - Test environments in a containerized testing world

Being able to set up isolated environments means that there is no real need for Development, Test, Acceptance and Production environments (DTAP) anymore. Instead, we can think in stages and use the quality gates within each stage to assess the quality level of our application and its components. In that case, we might actually say that DTAP is dead. Using the concept of environment-as-code, we can simply define an environment for a certain quality gate, for example performance test, set up that environment, and execute the required tests.

As we pass more and more quality gates, we are building up the trust required to actually run the application in production. In theory the order in which we pass the quality gates before production doesn't really matter. We could even run a few of them in parallel. But it is a good idea to consider the time tests take to run and what they actually test. For example, there is no real need to run relatively slow UI tests if your most basic smoke tests already fail. This might be conceptually similar to what you would do in your DTAP environments today, but you are no longer limited by the number of environments available or by the type of test you can run on a particular environment. This means you can focus on how to get rapid feedback and how to fail fast.



Figure 4 - Stages and quality gates instead of DTAP

"With his dedication and perseverance Cornell guides every client to success."

Marc Duiker about Cornell Knulst



How to get started?

Before you can use the concept of containerized testing, the application you are testing needs to be containerized. Actually, not only the application you are testing must be containerized, all other components required to run the specific test must also be containerized. And you will require some sort of test agent or test container that executes the tests for you. For UI testing you will need a container that runs headless UI tests, and for your test data you will need a database inside a container with the test data pre-loaded as a snapshot. If you connect to external services, it is a good idea to have stubs inside containers that can replace this dependency when you run your tests. There are many more examples of test components you can come up with, but the bottom line is that you will have to make sure that everything you require to run your tests is available in your containerized test environment.

Now that we have containerized the components of our test environment, we need to describe the set-up of the various test environments using the concept of environment-as-code. For example, we can use Docker Compose to define blueprints of the test environments that will be deployed in isolation. Docker Compose supports the use of multiple compose files that complement or overwrite each other. This means that we can have a main Docker-compose.yml file for our application, and a Docker-compose.integrationtests.yml file that adds the specialized testing container(s) and that (re)configures services to connect to the stubs instead of a real external service. We then tell Docker Compose to set up an isolated environment using the combined configuration of these two files. Moreover, we can have multiple combinations of compose files that configure different types of environments to run different types of tests.

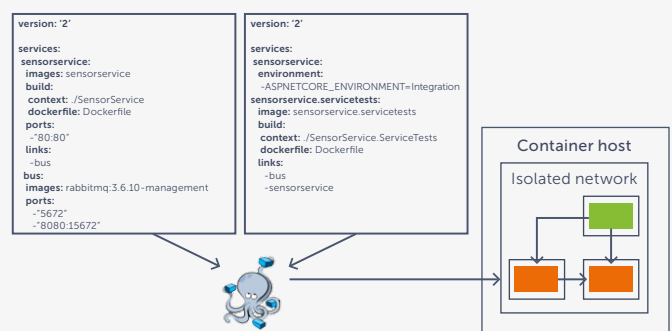


Figure 5 - Using multiple compose files to run your tests

Benefits

Partial test environments

One of the major benefits of containerized environments is the ability to set up partial environments. A partial environment means that you only have to include the services and test components required to run a specific type of test instead of your entire environment, for example only an API service and a back-end service.

Isolation

The containerized testing approach means that you can truly test your environment in isolation. There is no need to make your services accessible from the outside, because the agent running your tests is just another container running in the same environment as your services. For most types of tests there also is no need for your services to communicate with the outside world because your services talk to stubs instead of to the actual external services, and these stubs run in the same environment as your services. What's more, you don't have to worry about other running containers interfering with your test environment, unless you have explicitly configured it that way. Because you run your test agent within a container, you need a CI/CD orchestration tool (e.g. VSTS) to start your containerized testing environment. If you make use of Docker Compose, you can use the Docker compose up command to start your containers and the actual test execution.

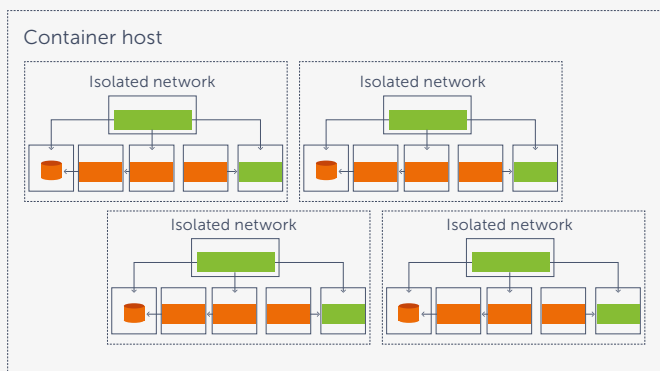


Figure 6 - On-demand and parallel test environments

On-demand parallel test execution

You can create your test environment exactly when you need it. You are now able to deploy your isolated environment on any container host and run multiple types of test in parallel, because you can set up an environment for each of them. You can now consider running security tests, performance tests and, for example, UI tests all at once. You don't have to wait for environments to become available so the overall execution time of your tests will decrease significantly.

Reduced costs

Using the container infrastructure, you can set up as many environments as you want. There is no real need for dedicated, pre-provisioned testing environments any longer; you can create a set of containers on a container host and run specific tests against them. Once you are done with

the tests, you no longer need the environment, so it can be destroyed. This means that you don't have any pre-provisioned environments that are idle most of the time. You still do need a container host, but you can utilize the available resource more efficiently than you ever could with separate pre-provisioned test environments.

Conclusion

There are already many practices available that cover different aspects of test automation. However, there are only few practices available for infrastructure when it comes to automated testing. Leveraging the same benefits that you get from containerizing your application for test automation allows you to look at infrastructure for test automation in a whole new way. The concept of a test environment changes from a pre-provisioned set of servers to an on-demand environment that contains everything needed to run just that specific type of test. This also changes the way you think about executing your automated tests. You don't have to think about DTAP anymore; instead, you can think about which types of test (quality gates) you need to run in order to build confidence to move to the next stage. Instead of thinking about which test has to run in which environment, you can think about which tests provide the fastest feedback.

Containerized testing is a great opportunity for organizations that are starting or thinking about containerizing their application stack. Containerizing your test infrastructure in addition to your application stack, speeds up your feedback cycles and accelerates your time to market. </>

Marco Mansi about Sander Aernouts

"Quick learner, hard worker, food lover, his ALM solutions are the best DevOps recipe for success!"



The Ultimate Education Destination

2017 Orlando

ROYAL PACIFIC RESORT AT UNIVERSAL ORLANDO
NOVEMBER 12-17



Live! 360: A Unique Conference for the IT and Developer Community

- 6 FULL Days of Training
- 5 Co-Located Conferences
- 1 Low Price
- Create Your Own Agenda from Hundreds of Sessions
- Expert Education and Training
- Knowledge Share and Networking

CONNECT WITH LIVE! 360



[@live360](https://twitter.com/live360)



facebook.com
Search "Live 360"



linkedin.com
Join the "Live! 360" group!

EVENT PARTNERS



Microsoft

Magenic

PLATINUM SPONSOR

SMARTBEAR

SUPPORTED BY



Visual Studio

msdn
magazine

Redmond
Channel Partner

NEW: HANDS-ON LABS



Join us for full-day,
pre-conference hands-on
labs Sunday, November 12.

Only \$695 through November 12



**REGISTER
NOW**

**REGISTER BY
NOVEMBER 12
AND SAVE \$300!***

Must use discount code LEB01

*Savings based on 5-day packages only.
See website for details.

Check out our other 2017 events
for Developers and IT Pros:

- Visual Studio Live! - vslive.com
- TechMentor - techmentorevents.com



TECH EVENTS WITH PERSPECTIVE

5 GREAT CONFERENCES, 1 GREAT PRICE

Visual Studio LIVE!
EXPERT SOLUTIONS FOR .NET DEVELOPERS

Visual Studio Live!: Code in Paradise at VSLive!™, featuring unbiased and practical development training on the Microsoft Platform.

SQL Server LIVE!
TRAINING FOR DBAs AND IT PROS

SQL Server Live! will leave you with the skills needed to drive your data to succeed, whether you are a DBA, developer, IT Pro, or Analyst.

TECHMENTOR
IN-DEPTH TRAINING FOR IT PROS

TechMentor: This is where IT training meets sunshine, with zero marketing speak on topics you need training on now, and solid coverage on what's around the corner.

Office & SharePoint LIVE!
ON-PREMISE, CLOUD & CROSS-PLATFORM TRAINING

Office & SharePoint Live!: Today, organizations expect people to work from anywhere at any time. Office & SharePoint Live! provides leading-edge knowledge and training to work through your most pressing projects.

ModernApps LIVE!
MOBILE, CROSS-DEVICE & CLOUD DEVELOPMENT

Modern Apps Live!: Presented in partnership with Magenic, this unique conference delivers how to architect, design and build a complete Modern App.



IaaS, Containers or Serverless?

Choosing the right cloud solution for your web applications.

Microsoft Azure offers several options for hosting your web-based applications, and it is difficult to choose the best option for your scenario. It might look easy to just take the simplest way of moving your infrastructure to Azure in an IaaS way, but this is probably not the best option.

There is no single silver bullet option that is always best. In this article, we will discuss the various options to consider when you choose to move your web workloads to the cloud.

Author Chris van Sluijsveld & Geert van der Cruisen

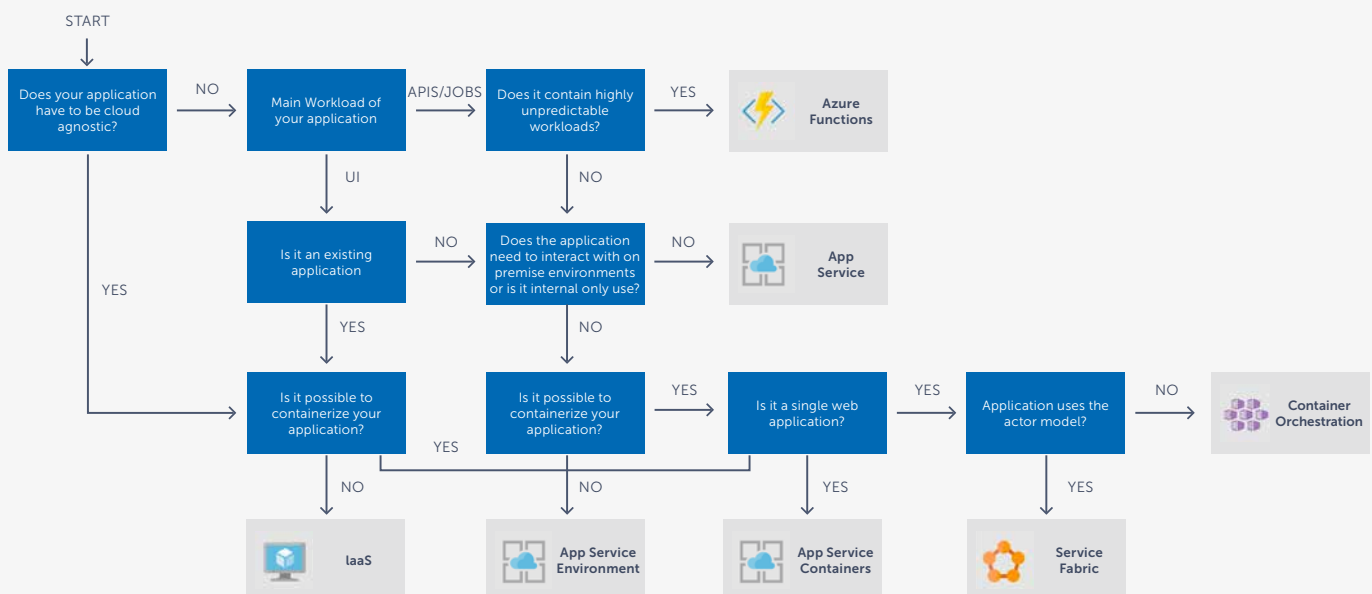


Figure 1 shows a flowchart that could help you decide which flavor fits your application. This flow chart focuses on several questions that we will elaborate on in the following section. Please consider that it is impossible to take all possible questions and scenarios into consideration in this flowchart, so use it as a rule of thumb. The questions described below are some of the main considerations you should take into account when choosing your cloud web infrastructure.



“You can tell Chris spends most of his time with his head in the cloud because he’s always smiling.”

Marc Duiker about Chris van Sluijsveld

Cloud Agnostic?

Microsoft Azure offers a lot of specific Platform as a Service features – for example App Service – that remove the effort of managing your infrastructure. When you choose a solution that is specific to Azure, it can become more difficult to make a switch to other cloud providers in the future. The choice is therefore between development speed combined with tighter coupling with Azure, or more overhead in managing your environments, which gives you more control over where you host your application. Typically, you’re fine with just choosing one cloud solution provider, but sometimes regulations, uptime requirements or a specific application architecture require that you use multiple providers to host your applications.

Containerizing applications?

Being able to host your applications using container technology such as Docker gives you a lot of flexibility and options. Microsoft is making working with containers better each day by adding support for developing applications in both Azure and Visual Studio. Containers can run anywhere on any cloud platform or on-premise, making it a very viable option for both new and existing applications. Tools such as Image2Docker (<https://xpir.it/xprt-right-cloud1>) make it easier to convert existing applications to Docker containers, so even legacy applications can be hosted using container technology.

Highly unpredictable and flexible workloads?

This question focuses on the usage of your jobs or APIs. In most scenarios, the workload on your APIs will be distributed quite evenly. When your workloads are very unpredictable or they tend to increase at certain moments while at other times there is no load at all (for example monthly batch calculations), specific solutions make a better fit than others. In Serverless options such as Azure Functions, you only pay for the actual use of the executed code, making it ideal for scenarios like this.

Interacting with on-premise / hybrid cloud scenarios?

Some hosting solutions cannot be added to a VNet in Azure, making it a lot harder to securely connect these options to resources that are not hosted on Azure. This is something to consider when building applications that should live in a hybrid cloud scenario where you connect a cloud solution to on-premise services or databases. Azure has made flavors for most of these options that include VNet support. For instance, for App Service they have created the “App Service Environment” option. However, setting this up is more expensive and requires more work compared to the normal App Service.

IAAS


Infrastructure as a Service (IaaS) is an instantly available computing resource. It is provisioned and managed for you by the Cloud vendor. It enables full customizing of the VM after the initial provisioning. But because it is only a managed VM, you are responsible for keeping the VM healthy, which means that patching and security hardening are the user’s responsibility.

Pros

- > Lift and shift – a way of working you are already familiar with
- > Easy to move between cloud providers
- > Can host any technology, Windows, Linux, any webserver technology
- > Familiar way of working for most IT departments
- > Complete freedom of workload

Cons

- > You are still responsible for a lot of management yourself, e.g., patching OS and maintaining configuration
- > Often not cheaper than non-cloud hosting
- > Not very effective usage of hardware

A man with a shaved head, wearing a dark blue suit jacket over a light blue and white striped shirt, stands with his arms crossed. He is smiling slightly and looking towards the camera. The background is a vibrant, abstract painting with bold colors like yellow, blue, and red, featuring large white letters and a stylized face on the right side.

**"A real mobile authority,
on the Xamarin platform
as well as on some
pumping dance beats."**

Martijn van der Sijde about Geert van der Crujsen

Azure App Service

Azure App Service is the Platform-as-a-Service offering from Microsoft. It allows for rapid deployment of your applications in a wide range of languages and frameworks. It allows for rapid scaling and a fast setup time, from local development to Azure Cloud production in 10 minutes.

Pros

- > Easy to set up and get going.
- > Fully managed infrastructure. Only focus on the web application itself
- > A lot of supported programming languages: NET, Java, Node.js, PHP and Python.
- > Easy to scale up or out

Cons

- > No VNet support for secure connections to on-premise from Azure
- > Directly exposed to the internet. No direct control over inbound and outbound traffic

App Service Environment

The App Service Environment offers a completely isolated Platform-as-a-Service hosting environment. It allows you to host your web workloads in a dedicated environment. The most important feature from a customer perspective is that it allows deployments into a customer-defined Virtual Network. Because an App Service Environment is deployed into a VNet, customers have finely grained control over both inbound and outbound application network traffic, as well as connecting to on-premise resources using VPN or express-route.

Pros

- > Isolated environment
- > Deployed in a customer's virtual network
- > Fine control over inbound and outbound traffic to your apps
- > The ILB (Internal Load Balancer) ASE offers the option to use your own application gateway and firewall to expose your applications to the outside world

Cons

- > ASE adds complexity during the initial roll out
- > Costly option if you don't have a lot of applications
- > High startup costs

Containers in App Service

The Containers in App Service offers the option to deploy Linux-based Docker containers in the App Service model. It offers all the benefits of App Service, but with the added benefit that it enables you to use the standardized Docker format to deploy your application.

Pros

- > No management of infrastructure
- > Use the familiar Docker format to deploy

Cons

- > No VNet isolation
- > Only Linux-based deployments (PHP, NodeJS, ASP.NET Core)

Azure Container Services

Azure Container Services (ACS) enable the rapid deployment of a container hosting environment using open source tools. At the moment these consist of Kubernetes, DC/OS and Docker Swarm. ACS allows you to deploy your complex applications as containers using both Windows and Linux as the base OS. Because it is a container-based solution, it is technology-agnostic as well as cloud-agnostic.

Pros

- > Lots of different orchestrators to choose from
- > Multi Cloud solutions are viable
- > Technology stack independent

Cons

- > You still have to manage infrastructure with OS / security updates

Service Fabric

Service Fabric is also an orchestrator on the Azure platform like ACS. But it also offers an SDK that describes how applications should be written. There is a choice between reliable services and an Actor Model. It also allows for guest executables to run inside Service fabric. Microsoft is also working on bringing Docker Container support to Service Fabric. Microsoft Service Fabric is not Azure only, you can install it on-premise as well as in other clouds.

Pros

- > Stateful services are provided out of the box using the Service Fabric SDK
- > Can run anywhere

Cons

- > Higher startup costs compared to other container orchestrators

Serverless

Serverless technology is the most cloud-native option in the list. Azure Functions and Logic Apps were made to offer you the things you expect from the cloud. In Azure Functions, you build a piece of code that is a single function with some input parameters and some outputs. Then this single function is uploaded to Azure and it will run from there. There are almost no operations involved and there are several triggers to start your function, ranging from HTTP triggers to timer triggers, to triggers that act on changes in table storage. Azure will automatically scale your function so it can run in parallel in order to quickly execute batch jobs or just single events, and the best part is that you only pay for each time the function runs.

Pros

- > Pay for what you use
- > Large set of triggers to start jobs
- > No infrastructure to manage
- > Easiest operation model

Cons

- > Not made for hosting web applications, only focused on APIs and jobs
- > No VNet support

Conclusion

As you can see, there are several options to host your web workload on Azure. We believe the days of IaaS are numbered, because in almost all scenarios PaaS or container options are superior. Microsoft is heavily investing in these areas in comparison with IaaS. Please note that the flowchart we created for this article is just a general rule of thumb. There can be other factors that may be important in your specific use case when deciding which solution fits your application. It is a good practice to keep the infrastructure in mind when designing your software architecture. Modern software architectures such as microservices are flexible in the way you host them and can fit in the cloud infrastructure solution you choose to implement. This also allows you to start small and simple, and adapt while your application or load increases in size. </>

Containers as a Service in Azure

You have heard about containers, and you realize the benefits of using them. But how do you get started? Sure, you can use Azure Container Services, but that has a number of downsides. For example, it deploys Virtual Machines, machines that you need to manage yourself. Wouldn't it be nice if you could host your containers in the Cloud, at scale, without having to worry about the underlying infrastructure? With Azure Container Instances, you can.

Author Loek Duys

What are Azure Container Instances?

Azure Container Instances (or ACI) consist of a PaaS service that was recently added to Azure. It offers the capability to run both Linux and Windows containers. By default, it runs single containers, which means that individual containers are isolated from each other, and cannot interact with each other. The isolation between individual containers is achieved using Hyper-V containers. This gives you the same level of protection as using a Virtual Machine would, by running your container inside a small utility VM. You specify the amount of memory in gigabytes for each container, as well as the count of CPUs to assign. Furthermore, containers are billed per second. This means you are in complete control, and that you don't pay for resources you don't need.

By assigning a public IP address to your container, you make it accessible from the outside world. If you look at Figure 1, you can see a container that exposes port 80. The port is connected

to a public IP address that accepts traffic at port 80 of the virtual host. Note that in ACI, port mappings are not available, so the same port number must be used by both container and container host.

Note *Container Instances currently expect containers that are always active. Task-oriented containers are not supported, as they will be continuously restarted automatically after they exit. Simply running `cmd` or `bash` on its own is not a long running process. Running a process such as ASP.NET Core, a Windows Service or Linux daemon does work.*

No more VM management

It is important to know that in ACI you don't need to own a Virtual Machine to run your containers. This means that you don't need to worry about creating, managing and scaling them. In Figure 1, both the network and the Virtual Machine - the container host - are completely managed for you, and you have no control over either of them.

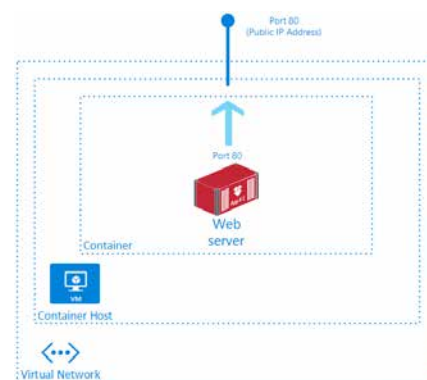


Figure 1: Azure Container Instances

Note *Running one container in ACI for a full month, with 1 CPU and 3.5GB of memory, costs about 30% more than running the same workload on a DS1 Virtual Machine running Linux.*

Running a container

The quickest way of creating a container in ACI is by using the azure command line interface (CLI 2.0):

```
az container create
--resource-group acidemo
--name logging
--image loekd/azurecontainerinstances.
logging:4.0
```


With the 'create' -command, you must specify the resource group to deploy into, the name of the container, and the image to base the container on. Optional parameters include credentials to your private registry, environment variables that can be used to inject configuration, the type of IP address to assign – public or private – and the port to expose.

Other important commands are:

- > 'logs' - this is used to show the logs (console output) of a running container.
- > 'show' - this is useful for debugging. This command displays the Resource Manager configuration container group, including a list of 'events' that contain logging information from ACI. If your container fails during deployment, this will show up here.
- > 'delete' - this is used to delete an existing deployment.

Note The output of 'az container show' command also includes configured environment variables in plain text. If they contain secrets, e.g., connection strings, they will be visible here. A safer way to inject secrets is by mounting a volume that contains the secret. The volume storage account key is not displayed.

```
"resources": [
  {
    "type": "Microsoft.ContainerInstance/containerGroups",
    "apiVersion": "2017-08-01-preview",
    "properties": {
      "containers": [
        {
          "name": "[parameters('loggingContainerName')]",
          "properties": {
            "image": "[parameters('loggingContainerImage')]",
            "resources": {
              "requests": {
                "cpu": 1,
                "memoryInGb": 1
              }
            },
            "volumeMounts": [
              {
                "name": "[parameters('volumeName')]",
                "mountPath": "/aci/logs/"
              }
            ],
            "ports": [
              {
                "port": 8080
              }
            ]
          }
        }
      ]
    }
  },
]
```

Figure 3: ARM Template, part 1

Now, if you open the Azure Portal and navigate to the resource group you specified, you will see your container running inside a container group.

Container Groups

By default, your containers are isolated from each other. But what if you need to have interaction between containers? To support this kind of scenario there is the concept of 'container groups'. Containers inside a container group are deployed on the same machine, and they use the same network. They also share their lifecycle: all containers in the group are started and stopped together.

Note At this time, container groups cannot be updated. To change an existing group, you need to delete and recreate it.

Note Containers are always part of a container group. Even if you deploy a single container, it will be placed into a new group automatically. When using Windows containers, a group can have only one container, this is because network namespaces are not available on Windows.

In Figure 2, you can see the anatomy of a container group. Here, three

containers are working together to form a fully functional application:

1. The 'Job Generator' container runs a Web Server at port **80** that receives traffic from the outside world and queues jobs to an Azure Service Bus queue.
2. The 'Job processing' container reads jobs from the queue and processes them. This container runs a Web Server at port **8000**.
3. Finally, the 'Logging' container is used by the other containers to persist logging information to Azure Files. It runs a Web Server at port **8080**.

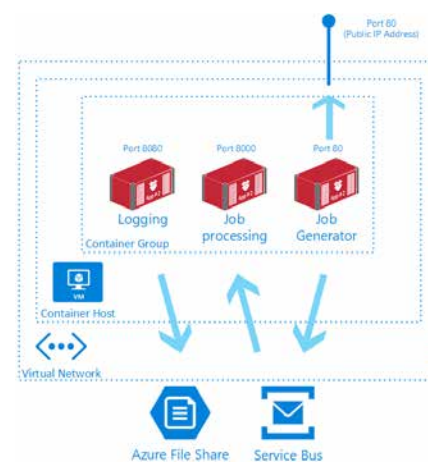


Figure 2: Container groups

Running a container group

A second way to create a container group is by using an ARM template. Figure 3 shows a simplified version of a template. In the array named 'containers', multiple containers can be described. You can specify the same options you saw earlier when using the command line. What is different, is that by using this approach, you can define multiple containers at once.

State

Any state you might store inside your container will be lost as soon as the container is removed. If you need to keep that state, you must store it outside the container. For example, if you generate files, you should store them on a file share. Fortunately, mounting an Azure File Share from ACI is very simple.

Using an Azure File Share in your container requires that you declare it in the `volumes` element, as you can see in Figure 4. To mount the volume,

```

"osType": "Linux",
"ipAddress": {
  "type": "Public",
  "ports": [
    {
      "protocol": "tcp",
      "port": "80"
    }
  ]
},
"volumes": [
  {
    "name": "[parameters('volumeName')]",
    "azureFile": {
      "shareName": "[parameters('shareName')]",
      "storageAccountName": "[parameters('storageAccountName')]",
      "storageAccountKey": "[parameters('storageAccountKey')]"
    }
  }
]

```

Figure 4: ARM template, part 2

the container configuration element `volumeMounts` references the volume by name. The value `mountPath` specifies the path at which the network share should become accessible. You can see this in Figure 3.

More about networking

To be accessible by the other two containers, the Logging container exposes port 8080. This port will be accessible within the entire container group, but not from the outside world and not from other container groups either. Containers within a group can communicate with each other by using 'localhost' and the exposed container `port`.

For example, calling the logging service API can be done by using the url: <http://localhost:8080/api/logs>. The platform will route traffic to the container.

Note Containers in a group are not discoverable through DNS. They can only be accessed through 'localhost', in combination with their exposed ports.

The Web server container exposes port 80. This port number matches the port specified to expose at the public IP address (`ipAddress`) of the container host. You can see this in Figure 4. The result of this configuration is that the two ports are connected. This makes the container accessible both from within the container group as well as from the outside world.

All containers in a group share one public IP address because they run on the same host. They also share a network namespace, so port mapping and port sharing is not possible. This means you cannot run multiple instances of the same container exposing the same port in one group. So if you require interactive containers to be load-balanced, or if you need to support upgrades without downtime, you'll need multiple container groups working together.

Seamless upgrades & Load balancing

The ACI platform will monitor both your containers and the container host. If there is a failure in either, ACI will attempt to correct this automatically by restarting the containers. At this time, ACI does not monitor your Container Registry for any image updates. This means that you'll need to upgrade deployments yourself. One way to do this is by running a container that monitors the container registry and triggers redeployment whenever a new image version is available.

For uninterrupted availability during upgrades and failures, you'll need to run multiple container groups and manage the network traffic flowing into them. This way you can replace one group, while others remain operational. You can see an example of this in Figure 5. In this example, Azure Traffic Manager is used as a router that directs incoming traffic to the two container groups.

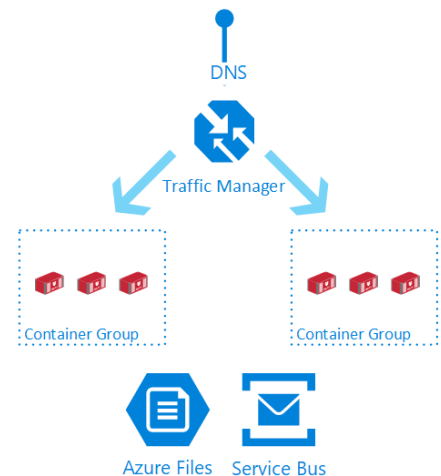


Figure 5: Using Azure Traffic Manager

To update one group, you would need to take the following steps:

1. Create new container groups that run the new version of the product, and add them in traffic manager.
2. Remove the old container groups from traffic manager.
3. After the DNS TTL has expired, take the old container group down.

Doing this manually, and perhaps even multiple times a day, is error-prone and frankly, it is a waste of time.

Container orchestration

If you just need to run a few containers without having strict requirements for availability, ACI is a great platform. Applications that are more mission-critical are likely to have more requirements, for example controlled, automatic, seamless application upgrades. This is very difficult to build yourself and it probably is not your core business.

Fortunately, there are products that can make life a little easier. This brings us back to Azure Container Services (or ACS). Kubernetes is one of the three container orchestration platforms offered by ACS. It works by smartly combining a group of Virtual Machines – called agent nodes – to form one virtualized container hosting platform. Agent nodes can be added and removed on the fly in order to deal with varying usage loads. Agents can be either Windows-based or Linux-based machines. The Kubernetes platform can be used to deploy, run, monitor, scale and upgrade containerized workloads.

In Kubernetes, these workloads are called deployments.

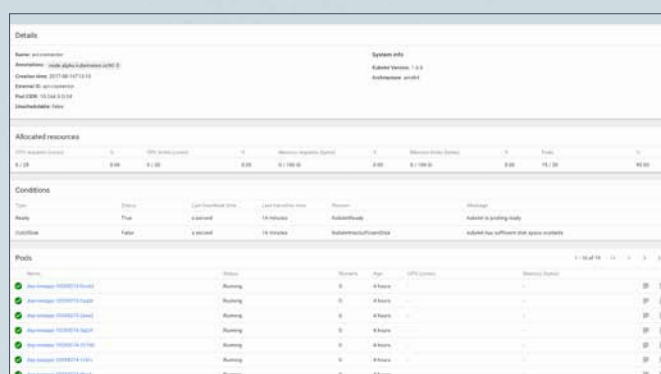
Best of both worlds

Kubernetes agents are Virtual Machines that require maintenance, and it would be great if you could replace those agents with ACI. Well, it turns out you can. You can do this by installing the ACI connector for K8S, which is an open source initiative on GitHub: <https://xpir.it/xprt5-caas1>.

After installing this connector on a Linux node in your cluster, an additional agent node will appear in your Kubernetes cluster. The connector will create and delete container groups on the fly. The new agent can run **both** Windows-based and Linux-based containers. When using Linux-based containers, you can create deployments using multiple containers; they will translate into container groups in ACI.

Because Kubernetes performs rolling upgrades for you, deployments to ACI become much easier. Using Kubernetes to orchestrate your containerized workloads enables your team to deploy to production several times a day. Combining Kubernetes with Azure Container Instances enables you to run Linux-based and Windows-based containers, allowing you to choose the best technology stack for each container you use. At the moment, this is the closest you can come to running Containers-as-a-Service in Azure.

For a complete, working sample implementation of three containers running on the ACI platform, please visit my GitHub project at: <https://xpir.it/xprt5-caas2>. </>



The screenshot shows the Kubernetes dashboard. At the top, there's a 'Details' section for the 'aci-connector' node, showing its name, namespace, creation time, and status. Below this is the 'Allocated resources' section, which shows a table of resources allocated to the node. The 'Conditions' section shows the node's status as 'Ready'. At the bottom, the 'Pods' section shows a list of pods running on the node, including the ACI connector itself and several application pods.

Details	
Name: aci-connector	System info
Namespace: kube-system	Kubernetes Version: 1.9.0
Creation time: 2017-05-17 17:10:10	PodTemplate: acik8s
Resource ID: aci-connector	
PodTemplate: aci-connector	
PodTemplate: aci-connector	

Allocated resources	
Resource	Quantity
aci-connector	1
aci-connector	1
aci-connector	1
aci-connector	1
aci-connector	1
aci-connector	1
aci-connector	1
aci-connector	1

Conditions	
Type	Status
Ready	True
Ready	False

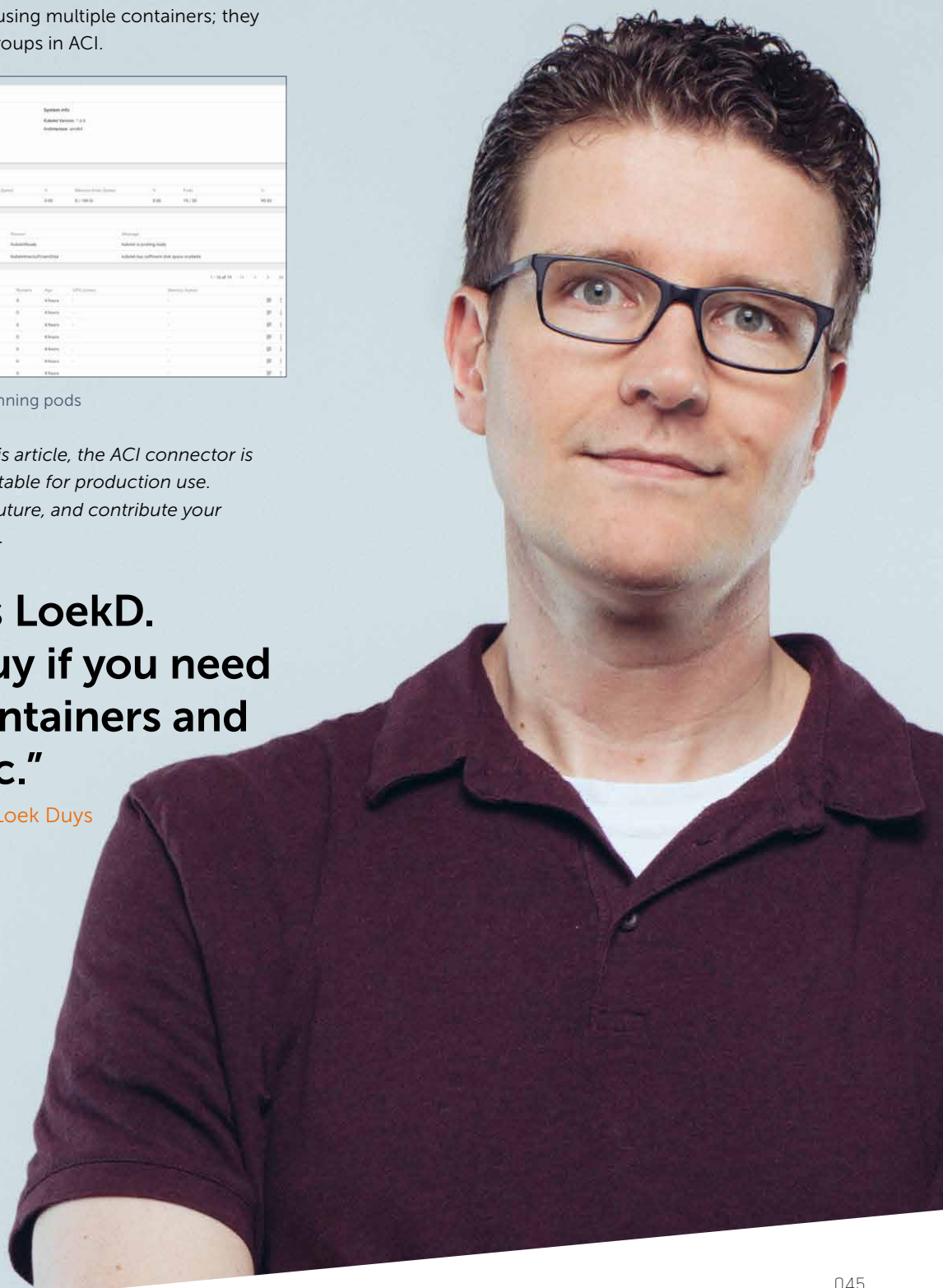
Pods	
Name	Status
aci-connector	Running
aci-connector	Running
aci-connector	Running
aci-connector	Running
aci-connector	Running
aci-connector	Running
aci-connector	Running
aci-connector	Running

Figure 6: ACI-connector node running pods

Note At the time of writing this article, the ACI connector is still in preview and not yet suitable for production use. Do keep an eye on it for the future, and contribute your improvements to the product.

**"The famous LoekD.
The go-to guy if you need
help with containers and
service fabric."**

Chris van Sluijsveld about Loek Duys



Best practices using Azure Resource Manager templates

This article focuses on best practices regarding the automated deployment of resources to Azure. We have implemented Continuous Deployment (CD) pipelines including the provisioning of Azure resources for many customers, and we would like to share our experience so you can benefit from it. These practices will help you create more reliable, testable, reusable, and maintainable templates.

Author Peter Groenewegen & Pascal Naber

Automate deployments to Azure

Azure Resource Manager templates (ARM templates) are the preferred way of automating the deployment of resources to Azure Resource Manager (AzureRM). ARM templates are JavaScript Object Notation (JSON) files. The resources that you want to deploy are declaratively described within JSON. An ARM template is idempotent, which means it can be executed as many times as you wish, and the result will be the same every time. Azure takes care of the execution and identifies the changes that need to be executed.

When provisioning infrastructure, we apply the same best practices as with deploying applications. This is also known as Infrastructure as Code¹. Applying CD enables you to develop your infrastructure in a repeatable and reusable way, and you can reuse your ARM templates over multiple teams by applying these practices. This allows you to use a dashboard to monitor the quality of the infrastructure provisioning.

To execute ARM templates in a CD pipeline, our preferred method uses Visual Studio Team Services (VSTS). The execution is being done by a VSTS task: "Azure Resource Group Deployment". Make a VSTS dashboard to monitor all your builds and releases, and this will give you a quick overview of the state of your environments and the quality of your templates. It is very useful to show your team and other stakeholders what you are doing.

Use linked ARM templates

When you create an ARM template for multiple resources, for instance for a whole project environment, ARM templates allow you to declare multiple Azure resources in one ARM template (Figure 1). For example you can create one big ARM template for a Storage Account, Azure SQL, Azure Web App and an Azure Redis Account. Although this is technically possible, we have learned from experience that it's a good practice to declare each resource in its own separate ARM template.

In effect, we apply the Single Responsibility Principle for Azure Resources in our Infrastructure as Code solution. In this sample, we would create one ARM template for the creation of a Storage Account, one ARM template for Azure SQL, one for the Azure Web App, and one for the Redis Account. But how can you glue the resources together so that the result is the whole project environment, and deploy it in one step? The answer is using Linked ARM templates².

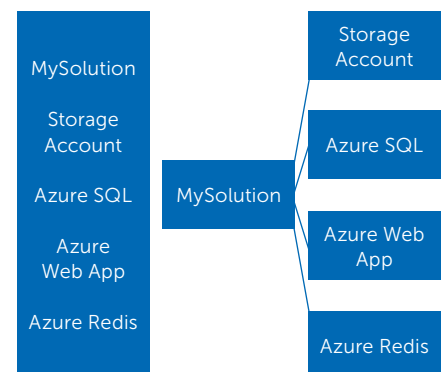


Figure 1: A single ARM template with all resources versus a Linked ARM template linking to separate ARM templates per resource type.

¹ Infrastructure as Code [<https://xpirt.it/xprt5-arm5>]

² Resource group linked templates [<https://xpirt.it/xprt5-arm6>]

Linked ARM templates enable you to link from one template to another template. Linking templates enables you to decompose your templates into purpose-specific reusable parts. These parts are more readable, testable and reusable than when you would copy them to other deployment templates. The main template links to multiple sub-templates, which can be another composition of sub-templates or actual resource templates themselves. The composition of the resources hides the implementation details of the underlying resources by only exposing an interface by means of the parameters that can be changed in the customer environment.

An example of a linked template:

```
"resources": [
  {
    "apiVersion": "2017-05-10",
    "name": "linkedTemplate",
    "type": "Microsoft.Resources/deployments",
    "properties": {
      "mode": "incremental",
      "templateLink": {
        "uri": "https://www.contoso.com/AzureTemplates/myLinkedTemplate.json",
        "contentVersion": "1.0.0.0"
      },
      "parameters": {
        "myparameter": { "value": "myparametervalue" }
      }
    }
  }
]
```

The provisioning of a resource group normally has multiple high-level resources and their sub-resources (Figure 2). In practice, you'll get a main template which references multiple composed templates. These composed templates reference lower level composed templates or the actual resource templates that create resources. In most cases, at the leaf level of this tree hierarchy, a template only contains one resource. These leaves are easy to test by themselves and they are reusable in multiple compositions.

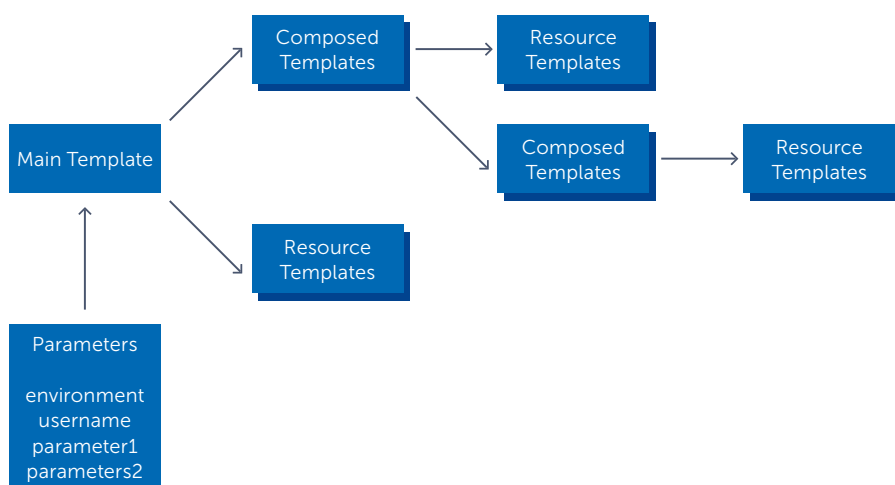


Figure 2: Breakdown of linked ARM templates

Apply T-Shirt sizes

T-Shirt sizes are named configurations which have proven to be working. The name encapsulates the underlying sizing of the Azure Resources. The consumer of the ARM template can only choose known working configurations. This will also prevent configuration errors and it saves time when creating new templates.

³ World Class ARM templates - Considerations and Proven Practices, June 30 2015, Marc Mercuri, Principal Program Manager | Ulrich Homann, Distinguished Architect | George Moore, Principal Program Manager Lead [<https://xpirt.it/xprt5-arm7>]

⁴ T-shirt sizing arm templates [<https://xpirt.it/xprt5-arm8>]

Usage of linked templates

The `http` or `https` address in the `templateLink` property has to be accessible for the Resource Manager service to execute the ARM template. When you make use of a publicly available web address, this is easy. But if you don't want to make your underlying ARM templates available for everyone, it's easy to store the ARM templates in a private container in Azure Storage and access the ARM templates with a SAS token.

VSTS task to create a SAS token
<https://xpirt.it/xprt5-arm1>

In case you are provisioning Virtual Machines (VM), the ARM templates can offer a wide range of possibilities in configuration combinations. All input parameters of the Virtual Machine (VM) template can be exposed. Now you could create combinations of resources that are not supported by AzureRM. For example, a DS VM with Standard storage is not allowed.

A good way to manage different sizes of resources, which also minimizes the various configuration types, is using known configurations, also called T-Shirt sizing³. For example, a T-Shirt size is an abstraction, like a Small, Medium or Large version of a VM⁴. The sizing hides the underlying real sizes of the created resources. For example, a Small T-Shirt size would deploy a combination of small resources of the underlying templates. You can test the various T-Shirt sizes, which prevents creating combinations that are not working. This also takes away the complexity of the underlying resource combination from the consumer of the template.

Test every template and automate it

Testing your templates helps you to maintain your quality. Testing the ARM template is done by executing the ARM template. This is the smallest possible unit that can be provisioned. It also provides samples for the consumers of your templates. Create at least one test deployment for each ARM template you have.



TECHORAMA

DEEP KNOWLEDGE IT CONFERENCE



A new conference experience is
coming to the Netherlands

———— AUTUMN 2018 ————

TECHORAMA.NL

@TechoramaNL

Run the tests on each new version to see whether the ARM templates are working. The smaller the tests, the more specific and faster it is to nail a specific issue, e.g. typos and misconfigurations. This will save you time in the long run.

Use output parameters

When you want to use the connection string of the Azure SQL database in the Azure Web App, you need to use an output parameter in the ARM template of the Azure SQL resource. This output parameter contains the connection string, which is an input parameter in the Azure Web App. The Linked ARM template glues the output of the Azure SQL template together with the input of the Azure Web App.

This will make the templates less prone to errors. When you change something in the Azure SQL Template, all depending templates will get the new value because they reference the output parameters of the first template.

Another benefit is that the order of execution is automatically recognized by AzureRM, based on these output parameters. Templates which are set up like this don't need a dependsOn property to specify dependencies. This way you don't have to interpret the sequence of execution of ARM templates yourself. We consider the usage of dependsOn an anti-pattern in a linked ARM template.

Output parameter:

```
...
"outputs": {
  "myResourceName" : {
    "type" : "string",
    "value": "[reference(resourceTemplate).name)]"
  }
}
```

Usage of output parameter:

```
"parameters": {
  "myparameter":{"value": "reference('myResourceWithOutput').outputs.
myResourceName.value"}
}
```

Make a naming convention template

The first discussion that always comes up during a workshop on applying CD to Azure resources is the naming convention for these resources. After finding out how you want to name resources, you have multiple possibilities. When you pass

```
{
  "$schema": "http://schema.management.azure.com/schemas/2015-01-01/
deploymentTemplate.json#",
  "contentVersion": "1.0.0.0",
  "parameters":{
    "shortDescription":{
      "type":"string",
      "maxLength":2
    },
    "shortEnviroment":{
      "type":"string",
      "maxLength":1
    },
    "location":{
      "type":"string",
      "maxLength":2
    }
  },
  "variables":{
    "nameconvention":"[concat('myname',parameters('shortDescription'),
parameters('shortEnviroment'),parameters('location'))]"
  },
  "resources":[
    ],
    "outputs":{
      "name" : {
        "type" : "string",
        "value": "[variables('nameconvention')]"
      }
    }
  }
}
```

the whole name of the resource that needs to be created using a parameter file, you'll find yourself creating duplicate code because the largest part of the name is defined by a standard pattern. Another way is to apply a naming convention, in which case you only need to pass parts of the name. For example, the ARM template of Azure SQL generates the resource name based on the input parameters and pattern. But each resource has a slightly different name, which means that each ARM resource contains this pattern. It often happens that the naming convention changes. In both cases you must change either all your ARM templates or all your ARM parameter templates.

To apply the Single Responsibility pattern and to prevent duplicate code, we apply a naming convention template. We make use of the fact that ARM templates don't always have to create resources, but can do other things as well. In the naming convention template, no resources are declared. It does make use of input parameters and the name of the resource is returned as an output parameter.

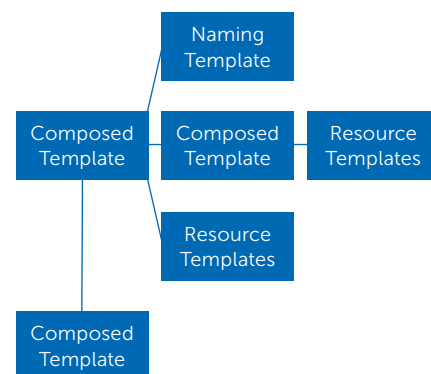


Figure 3. Using a named template in a composed template

Minimize the number of parameters

When using linked templates, make the number of parameters the smallest set possible. Do this by only parameterizing the variables that are different over your environments. This way you keep the input of your templates small and changes that can cause misconfiguration become less likely.

Don't misuse the default value for parameters

Don't set default values for required input parameters or parameters that need to differ over environments. We often see the misuse of default values. The default value is chosen for a single deployment to a single environment without thinking about future deployments. If you want to pass a value, you must use a parameter file.

One deployment per resource group

Another question that comes up very often is how to apply a logical division of resources over resource groups. Technically it is possible to deploy all resources to a single resource group. In practice, it is handy to provision resources with the same lifecycle grouped into the same resource group. Many side services of your application have a different lifecycle from the application itself, for example the data, logging, authentication, networking, etcetera. When you remove your application from Azure, these side services will still exist.

An ARM template is executed on a single resource group by default and this is also considered a best practice. An application can be deployed to multiple resource groups. Each resource group has its own ARM template with resources. Manage your resource group based on the lifecycle of the resources in that resource group.

Keep secrets out of your deployment parameters

There are multiple ways to keep your secrets out of your deployment parameters so developers do not have to know or see the secrets. If you are using parameters for secrets that can be provided by Azure resources, then use output parameters to 'stream' the secret directly to the resource that needs it. For example, if you are using an Azure SQL connection string as a parameter in your current deployment, you can get the connection string directly from the Azure SQL Database by using an output parameter. If you have other secrets,

such as ssh keys, disk encryption keys, passwords, etcetera, you have the possibility of using variables in VSTS and mark them as secret. For Azure WebApp web settings, you can use a VSTS task Azure Web App Configuration to apply (secret) variables to your WebApp⁵, or for other resources provide the secrets by using the Azure Key Vault.

The person who is responsible for creating the templates does not have to know secrets in the provisioned environment, for example by putting the sensitive data into the Key Vault. If you grant your deployment pipeline access to the Key Vault, the developers don't have to know the secrets in the Key Vault. Read how you can apply the Key Vault to your deployments here:⁶.

Use a complete deployment mode as much as possible

When deploying resources to a resource group, complete deployments will guarantee that your resources in the resource group are the same as in your source control.

When you manage your resources in AzureRM with ARM templates, you have three options for execution:

1. Validate
2. Incremental (default)
3. Complete (advised)

Validate means that the AzureRM validates the template. This can be useful to see whether a change in a template or variable passes the basic validation. The template is compiled, but not applied on Azure. This can be done as a first step in a provisioning pipeline. After validation, you know that the syntax is correct, simply because it compiles. It can still fail when you execute the template, however you have checked the schema and syntax before execution of the template.

Incremental is the default mode. It only deploys the new resources in the ARM template. No resources are removed. So, if you have renamed an SQL Server database, the database you created earlier will still exist after applying the ARM template with the new database.

Tips and tricks

Assign tags

Tagging your resource will help you with organizing your resources.

A tag adds a custom property to a resource that can be queried later.

Some samples of tags that can help you:

- > Billing (Cost Center tag)
- > Which department
- > Environment

Use tags to organize your Azure resources

<https://xpirt.it/xprt5-arm2>

Prevent accidental deletion of resources

Locks help you prevent users or templates deleting resources by accident. If you have crucial resources in your deployment that should not be deleted, a lock prevents accidental deleting. A lock can only be removed from the portal or PowerShell.

Lock azure resource to prevent accidental deletion

<https://xpirt.it/xprt5-arm4>

Assign policies

Policies can help you with establishing conventions for resource deployments. A resource deployment that does not follow the conventions will be rejected.

Some building policies are:

- > Allowed locations
- > Allowed resource types
- > Allowed storage account SKUs
- > Allowed virtual machine SKUs
- > Apply tag and default value
- > Enforce tag and value
- > Resource types that are not allowed
- > Require SQL Server version 12.0
- > Require storage account encryption

What are resource policies

<https://xpirt.it/xprt5-arm3>

⁵ Web app settings configuration <https://xpirt.it/xprt5-arm9>

⁶ Keep your deployment secrets in the key vault [<https://xpirt.it/xprt5-arm10>]

A man with short, spiky brown hair and blue-rimmed glasses is smiling at the camera. He is wearing a blue and white checkered button-down shirt and a brown leather belt. His arms are crossed. He is standing in front of a wall covered in colorful graffiti, including the words 'APPLE', 'WARE', and 'RUNNING'.

**"Pascal will kill
your Monolith
and unleash the
power of Azure."**

Peter Groenewegen about Pascal Naber


```
format if (numOfdot == -1):
    string.replace("czFieldID", str(in
    string4replace + tempString
    ate_type_buffer tempString =
    string4replace + tempString
    string.replace("czDataType", "B
    "KT"): for line in searchlin
    ONTDHRicName = searchObj1.gro
    "KT" + ONTDHRicName + "\t" + opaque
    ONTDHRicName = "" opaqueV = ""
    ff) import shutil if os.path
    .splitlines() for line in con
    obj.group(1)] = searchObj.gro
    search(r'(\w+)\.', str(fNa
```

Idempotent

The execution of an ARM template is an idempotent operation. The execution of the same template should result in the same result each time. Only the required changes are applied to AzureRM. When building and executing ARM templates this property should be used in your advantage.

Complete executes the template and applies the template idempotently. When finished, only resources that are defined in the template are in the resource group. This way your AzureRM resources are managed from your template alone. If you remove a resource from the template, it will also remove that resource from AzureRM. For example, when removing Network Security Groups or Firewall rules from a template, the resources will be removed when you execute the template. All resource management is done from the ARM templates (Infrastructure as Code). The best practice is to deploy your ARM templates in Complete mode.

All ARM templates should be able to be executed idempotently, but unfortunately, not all ARM templates can be executed idempotently. This is considered a bug. You can check whether there is a newer version of the API in which this has been fixed. If the ARM template is not idempotent and you configure your deployment as Complete, the provisioning can fail. In this case, mark the deployment as Incremental and isolate the incremental part.

Conclusion

When you have embraced Azure for your applications, you should have a CD pipeline to take full advantage of the cloud. This CD pipeline should also contain the provisioning of the Azure resources you are using. Microsoft makes it possible to provision resources with ARM templates, but you have to spend time to create these ARM templates. Provisioning AzureRM with ARM templates will result in a more maintainable and reusable way of managing the resources. The best practices in this article have a learning curve, however, in the end you will be able to manage your resources more reliably and in less time. Setting up deployment pipelines for the resources, also known as Infrastructure as Code, is an investment you will benefit from in the long run. The result is that you'll never have to access the Azure portal to add or change resources anymore and you are fully in control over what is deployed in your Azure Resource Manager environment from source control. For sample ARM templates, see the Enterprise Application Template gallery on Github⁷. </>

⁷ <https://xpirt.it/xprt5-arm11>

BIG DATA SURVEY 2017.

HOW EUROPEAN ORGANIZATIONS IMPROVE
WITH SMART DATA APPLICATIONS.

FREE REPORT BIG DATA SURVEY

Over 800 European organizations participated and shared their insights and experience about how they leverage data to increase their competitive edge.

Download the in-depth Big Data Survey report now on www.bigdatasurvey.nl



**DOWNLOAD THE
FULL REPORT WITH
IN-DEPTH ANALYSIS
AND INSIGHTS IN
HOW TO RECRUIT
DATA PROFESSIONALS,
CLOUD SECURITY,
RUNNING EXPERIMENTS,
THE MOST POPULAR
TECHNOLOGIES AND
MUCH MORE.**

WWW.BIGDATASURVEY.NL

60%



60% of all participants indicate that their organization has a lot of potential with data. For 42%, data is part of their overall strategy.

The European data protection regulation GDPR takes effect on May 25, 2018. Only 1 in every 2 retailers say: "We're GDPR-ready!"



24%

Biggest challenge of becoming data driven?

1. Building up knowledge of data science & big data (**47%**)
2. Making time available for experiments (**45%**)
3. Support from the management (**33%**)



On average, organizations rate their current level of knowledge at 6 on a scale of 1-10.

6-



More than 54% of all participants see improving data quality as the biggest challenge of setting-up data infrastructure.

54%



Only **9%** of all websites use real-time personalization.

46% of all organizations use no cloud technology at all.

46%

GoDataDriven applies cutting-edge technology to accelerate your organisation along its data-driven path. Combined, our team has decades of experience with amazing technologies like AI, deep learning, big data, cloud and scalable architectures. We can help you kickstart or scale up your data-driven business, just as we have done at customers like:



GoDataDriven
proudly part of Xebia Group

Deployment pipelines for versioned Azure Resource Manager template deployments

Azure Resource Manager templates offer you a declarative way of provisioning resources in the Azure cloud. Resource Manager templates define how resources should be provisioned.

When provisioning resources on Azure with Azure Resource Manager, you want to be in control of which resources are deployed and you want to control their life span. To achieve this control, you need to standardize the templates and deploy them in a repeatable way.

This can be done by managing your resource creation as Infrastructure as Code.

Author Peter Groenewegen

Definition of Infrastructure as Code

Infrastructure as Code is the process of managing and provisioning computing infrastructure and its configuration through machine-processable definition files. It treats the infrastructure as a software system, applying software engineering practices to manage changes to the system in a structured, safe way.

<https://xpir.it/xprt5-iac>

The characteristics of Infrastructure as Code are:

- › Declarative
- › Single source of truth
- › Increase repeatability and testability
- › Decrease provisioning time
- › Rely less on availability of persons to perform tasks
- › Use proven software development practices for deploying infrastructure
- › Idempotent provisioning and configuration.

In this article, I will explain how to create Azure infrastructure with versioned Azure Resource Manager templates (ARM templates). For the deployments, the VSTS Build and Release pipelines were used. The code or ARM templates are managed from a Git repository. Code in the Git repository can use the same practices as any other development project. By updating your code or templates, you can deploy, upgrade or remove your infrastructure at any time. The Azure portal will no longer be used to deploy your infrastructure because all ARM templates are deployed by deployments pipelines. This will give you full traceability and control over what is deployed into your Azure environment.

Deployment pipelines for deploying versioned ARM templates

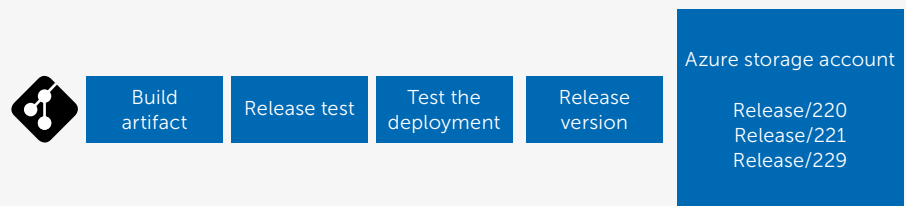
If you have a large number of infrastructure resources, it is good to know what the exact footprint is. If you know this, you can easily redeploy and create test environments without the constant question: to what extent is this infrastructure the same as production? In order to obtain adequate control over your infrastructure, you can apply versioning to the deployments and their content. In this case, ARM templates in combination with VSTS can help you. When applying Infrastructure as Code this way, you can test an actual infrastructure deployment and develop new templates at the same time. To do this you need two deployment pipelines:

- › Deployment of the reusable ARM templates (see the article on Best Practices Azure Resource Manager templates)
- › Pipeline for deploying the resource base on the reusable ARM templates.

The templates used in the second pipeline are deployed in the first pipeline. These are called linked ARM templates. A linked ARM template allows you to decompose the code that deploys your resources. The decomposed templates provide you with the benefit of testing, reusing and readability. You can link to a single linked template or to a composed one that deploys many resources like a VM, or a complete set of PAAS resources.

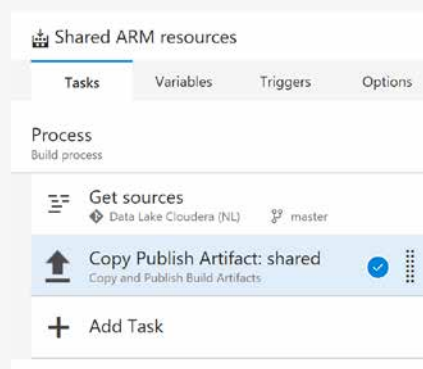
Deployment pipeline for reusable linked ARM templates

The goal of this pipeline is to deploy a set of tested linked templates (a version) to a storage account from where they can be used. Each time you perform an update to the templates (pull request), a deployment pipeline is triggered (continuous integration), and once all tests are successful, a new version is deployed and ready to use. The new deployments exist side by side with the earlier deployment. In this way the actual resource deployments can use a specific version. The following figure provides an overview of the pipeline:



Build

The deployment starts with a pull request to the master branch of the Git repository. Then a new build is triggered. In this build pipeline, the sources are copied to a build artifact to be used in a release. In addition, a build number is generated that can be used as version number of the released templates.

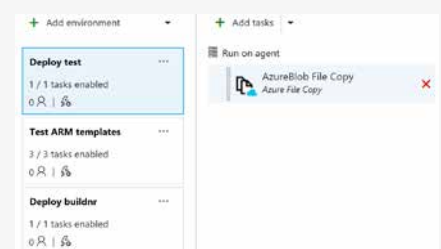


Release

The release has several release steps to ensure that the ARM templates are tested before they are published. Templates are tested by deploying the resources. In the sample, all steps are an automatic process. When the tests succeed, the release continues with the deployment of the templates to the storage account where they can be used for the real infrastructure deployments.

Deploy test

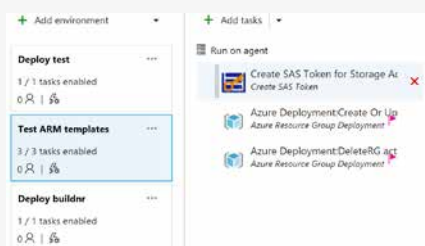
The first step is to deploy to a test location on the storage account. This test location will be used to test the ARM templates. When you deploy the templates, this can also help you in debugging errors by running a test deployment from your local machine. The only task in the environment is to do an Azure Blob File Copy. All the linked templates (the artifact) are deployed to the Azure storage account.



Test ARM templates

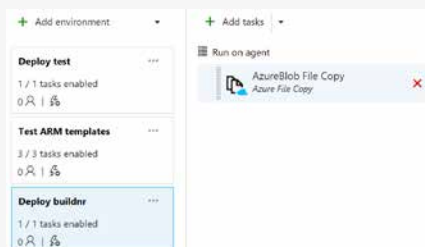
During the second step the ARM templates can be tested. First you get a SAS token (link: <https://xpir.it/xprt5-iac1>) to access the storage account. The next step consists of deploying the ARM templates. This runs a test ARM template that covers the parameters. When the deployment fails, the pipeline is stopped (asserted). The last step consists of removing the resource group where you have deployed your

test resources. If all steps succeed, the templates are approved for release. You can perform these steps multiple times for different types of resources and resource combination. When you perform this step N times, you can run them in parallel. If you split this into multiple templates, it is also clear where you have a problem if the step fails.



Deploy the production version

The last step will deploy the templates to a location where the build number is used in the naming convention. The task Azure Blob File Copy is the same as in the first step, only the location where the files are copied to is variable, depending on the build number. In this way the templates can be referenced by using the build number in the URL.



Pipeline for deploying the resource base on the reusable ARM templates

When all linked templates are deployed, they can be used to perform the deployments of your Azure infrastructure. In the sample pipelines, I have only one test environment, but the number of test environments can be different for each pipeline. One pipeline will deploy the resources of one Azure Resource group.

Build

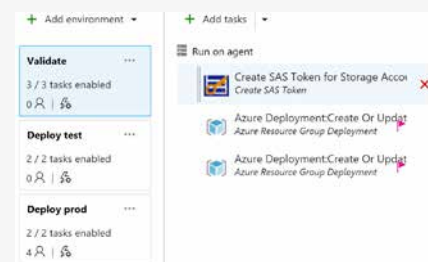
The goal of the build is to produce an artifact of the templates that can be used in the release pipeline. The deployment starts with a pull request to the master branch of the Git repository. Then a new build is triggered. In this build pipeline, the sources are copied to a build artifact to be used in a release.

Release

The release pipeline will validate, test and then deploy the resource to the production environment. The templates in each environment are the same; the only difference is the parameter files. The parameter file can parameterize the sizes of the resources deployed in the different environments. The sizes must be chosen wisely in order to represent the production environment, but keep in mind the costs of running a test environment. In the main template, you keep a variable build, which you use to point to a specific release from the previous pipeline. In this way you can control the deployed version of your shared linked templates.

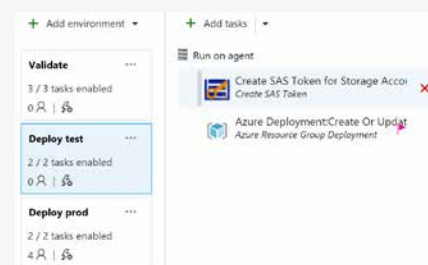
Validate templates

The first step consists of validating the templates for all environments. This is done by running a deployment in Validation Mode. Here, the template is compiled, it is checked to see whether it is syntactically correct and will be accepted by Azure Resource Manager. You have to do this for all environments to check whether the parameter files are correct. When the step succeeds, the deployment to test will start.

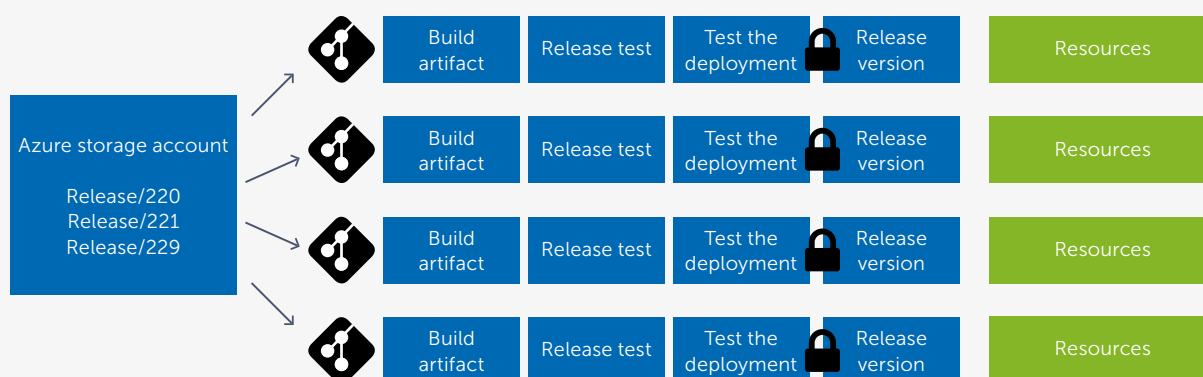


Release test

The goal of this step is to deploy and test the resource. If there is a need for a gatekeeper, approvals can be added at the beginning of this step. If not, the deployment of your test environment starts automatically. If possible, use the Deployment Mode option "Complete".



This ensures that the resources in the Azure Resource Group are the same as those defined in the ARM template.

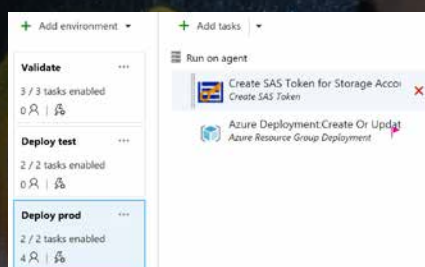


The action consists of 2 tasks.

First create a SAS token for access to the Azure Storage. The next step performs the actual deployment. If everything succeeds, you can optionally do some manual testing on the resource itself, or even add a script that does this.

Release production

The production release starts with a gatekeeper (approval). When you are satisfied with the previous (test) resources, an approver can start the production deployment. All resources in the production resource group will be updated according to the ARM template. Try to run your deployment in Complete mode, because then you know that all resources in the resource group are the same as you defined in your Git repository. You are running an infrastructure as Code scenario.



Final thoughts

Setting up deployment pipelines for your ARM templates is an up-front investment aimed at giving you control over the resources that are deployed in your Azure environment. When the pipelines are running, changes to your Azure environment are fully controlled from the code, and all changes are traceable from VSTS. When you have two staged pipelines, you have an Infrastructure as Code scenario in which you are in full control over what is deployed into Azure. </>

If he runs as fast as he writes C# code, he would beat Usain Bolt!

Marco Mansi about
Peter Groenewegen





**Cloud transformation
done right!**

We are Xpirit

Experts in new Microsoft Technology

www.xpirit.com/cloud

PROUDLY PART OF XEBIA GROUP



Think ahead. Act now.

www.xpirit.com

Think ahead.
Act now.



If you prefer the digital
version of this magazine,
please scan the qr-code.