TalkTalk innovates with a streaming TV service based on Azure Service Fabric

April 30, 2018

This post was authored by Ewan Fairweather, Mark Fussell and Gonzalo Ruiz from Microsoft in conjunction with Ilario Corna and Mike Goatly from TalkTalk

This is the first post where we will profile some of the customers who have been part of the Azure Service Fabric (https://azure.microsoft.com/en-us/services/service-fabric/) program over the last year and the applications they are deploying into production. We will look at why they chose to use Service Fabric and take a closer look at the design of their application, particularly focusing on a microservices design approach.

We kick off the series by looking at TalkTalk and their streaming TV service, TalkTalk TV.

In 2015, TalkTalk, the UK's third largest cable TV provider, acquired blinkbox, a popular movie and TV streaming service. Now rebranded and integrated as TalkTalk TV, they deliver the latest TV and movie content to a million monthly users, and growing, on Macs, PCs, game consoles, tablets, mobiles and smart TVs. Prior to the acquisition, blinkbox had already moved their complex video encoding and streaming application to Azure running in multiple regions to gain the flexibility of Azure's elastic compute and storage.

Blinkbox first migrated most of their existing application in a lift-and-shift fashion using 100 virtual machines (VMs) on Azure Infrastructure-as a-Service (IaaS). This approach enabled them to avoid rewriting code and run the application directly on those VMs (a detailed case study can be found here (https://www.microsoft.com/en-us/server-cloud/cloud-os/customer-stories/Blinkbox.aspx) with a video (https://channel9.msdn.com/Blogs/MSDNVideos/Azure-Case-Study-Video-blinkbox)).

However, with continued rapid growth and new business requirements following TalkTalk's acquisition, the TalkTalk TV team chose to redesign parts of their application to a microservices architecture using Azure Service Fabric.

Embracing scale, growth and continuous delivery

With more than 1.5 petabytes (PB) of data, TalkTalk TV uses Azure to host their catalogue of 30,000 items, which is delivered to millions of customers on multiple devices. In their initial IaaS implementation, TalkTalk used Azure blob storage (https://azure.microsoft.com/en-us/services/storage/blobs/) to store content spanning data centers in Amsterdam and Dublin for redundancy and used Azure Media Services (https://azure.microsoft.com/en-us/services/media-services/) to deliver the content from these locations.

After the acquisition, the team faced new business requirements and opportunities for scale and growth. They needed to delight existing TalkTalk customers with the breadth of content, and make it available through consumer devices and TalkTalk set-top boxes. They also needed to provide an expanded range of on-demand, catch-up TV content.

As TalkTalk began to integrate the newly acquired blinkbox capabilities, they took the opportunity to re-evaluate their design. Now the team needed to deliver features even faster and their content management and resolution platform would need to scale further to meet the projected requirements: the delivery of all content on demand to an additional 1 million TalkTalk customers directly from their set-top boxes.

The content and resolution workflow subsystem, which orchestrates the Media Services encoding and delivery, was built during the initial migration to Azure, using custom Windows services on IaaS VMs where the workflow state was stored in SQL Server. In the original design they chose to use database tables as queues to pass messages between services running on VMs. These VMs then communicated with Azure

Media Services to perform the necessary transcoding and protection of media. This approach required that they protect against lock contention, which introduced an unintended throughput issue in the workflow. The team found itself spending time implementing code to handle these edge cases and managing the custom services responsible for polling for work.

Talk Talk gained value from the resiliency, query processing and transactional capabilities of their SQL Server, and use it extensively for processing transactional data such as account updates and payments. However, in this specific case for media encoding they didn't need those query processing capabilities and found that using a relational database for their queuing and workflow system was not scalable. Furthermore, the use of database-stored queues to manage cross-service interactions required painstaking coordination to manage deployments and upgrades simultaneously every week and usually involved downtime of the application.

The advantages of Service Fabric and microservices

With the desire by TalkTalk TV leadership to move to more autonomous and experimental development practices, the team decided to break apart their monolithic application into microservices, and used the following principles to guide their technology selection:

- **Create independently deployable microservices: The n-tier architectures which had been moved to the cloud in an IaaS fashion made it difficult to add or modify functionality independently**
- **Autonomous experimental teams: The TalkTalk TV leadership desired to create nimble, teams that can deliver functionality quicker.**
- **Leverage a cloud platform: to minimize writing plumbing code, and maximize feature development, allowing the teams to ship more customer features.**

The TalkTalk TV team chose to build a new content management and resolution platform on Service Fabric, highlighting the following as key benefits:

- **Agility:** Rolling upgrades, granular versioning, packaging and deployment to achieve faster delivery cycles and maintain uptime during upgrades.
- **Development environment and programming models:** Developers were able to build and run the solutions locally using the Service Fabric SDK using a combination of the reliable actors and reliable services programming models.

- **Scalability:** The granularity of service scaling and versioning enables the system to scale as the number of users, devices, and content grows.

Designing the applications and services

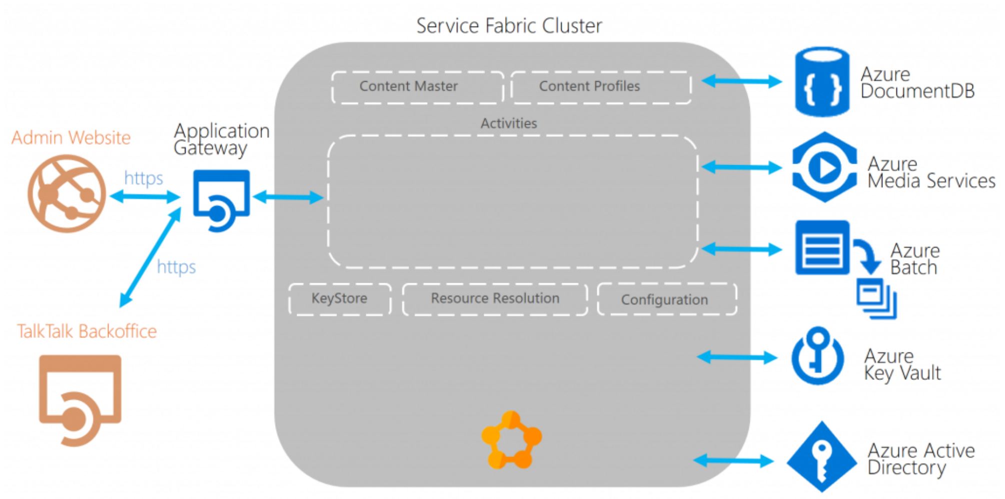
The TalkTalk TV team decided to break the encoding and resource resolution application into six Service Fabric applications using the following principles:

- **Applications are independently deployable and isolated:** A key advantage of the Service Fabric application and service model is the ability to update individual application instances and their services and move to a continuous deployment approach.
- **One Visual Studio solution per application:** Common libraries were distributed via a custom NUGET source, used to encapsulate common WebAPI, Service Fabric and logging logic.
- **Centralized configuration:** The team chose to have this in a separate application, where all other applications leverage this to get access to common configuration.
- **Endpoint per application:** Each Service Fabric application has at least one API that it exposes, where access to these is secured by authenticating against Azure AD, and separate applications were used to provide granular control of endpoint accessibility.

The diagram below illustrates the usage of a Service Fabric cluster to provide independently deployable and scalable microservices that coordinate with a number of Azure services. This shows how TalkTalk TV were able to gain benefits by re-writing only a portion of the IaaS application i.e. they were able to integrate in a service-oriented manner the new microservices with the existing admin website and TalkTalk back office system.

“ TalkTalk TV is a fast changing organization looking to embrace new and better ways of working whilst delivering the best customer experience. In a short span of time, Azure Service Fabric and the extended suite of Azure services has boosted agility, allowing the engineering team to implement outstanding quality microservices with a small number of developers”

—Ilario Corna: Head of Infrastructure
Content and Operations for TalkTalk TV



The TalkTalk content encoding and resource resolution application using Azure services

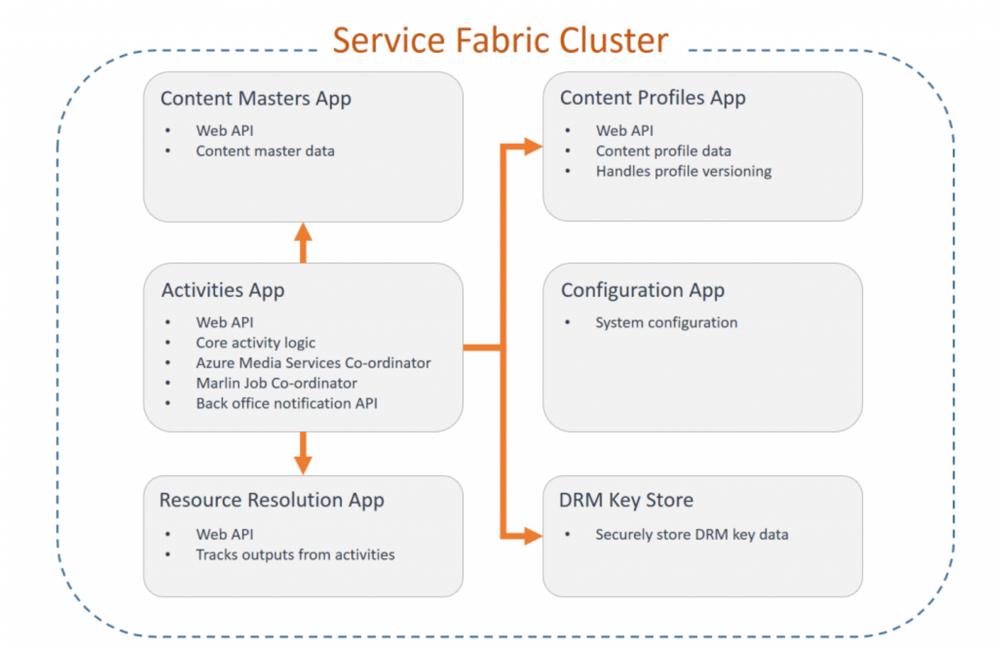
The following Azure services are used in the solution:

- **A Service Fabric cluster** hosts all the Service Fabric applications that make up the content encoding and resource resolution application. The Service Fabric applications consist of both stateless and stateful services.
- **Media Services** is used to encode and stream over 1.5 PB of content, encoded into a variety of formats that is served on demand to different devices.
- **Document DB** is used to store a JSON representation of the application's actor state. This enables disaster recovery, and is also used for ad-hoc querying.
- **Azure Application Gateway** (https://azure.microsoft.com/en-us/services/application-gateway/) is used to provide SSL termination management
- **Key Vault** is used to securely store application secrets, certificates and configuration keys.
- **Azure Active Directory** (https://azure.microsoft.com/en-us/services/active-directory/) is used to provide user and cross-service authentication and control access to the Service Fabric applications and their microservices.
- **Azure Batch** (https://azure.microsoft.com/en-us/services/batch/) is used to extend the Media Services capabilities and perform the Marlin encryption, a digital rights management (DRM) format. Batch provides a pool of VMs and the ability to orchestrate, submit, and monitor jobs. The workflow running in Service Fabric decides the type and number of VMs to use, then submits the job using the Azure Batch SDK, where the data is input and output via Azure blob storage.

To enable the content encoding and resource resolution workflow there are six different applications. The diagram below illustrates the roles and interactions between each of the separate Service Fabric applications.

“ The one-box development experience with the full Service Fabric services available locally increased our productivity and enabled us to deliver more features faster.”

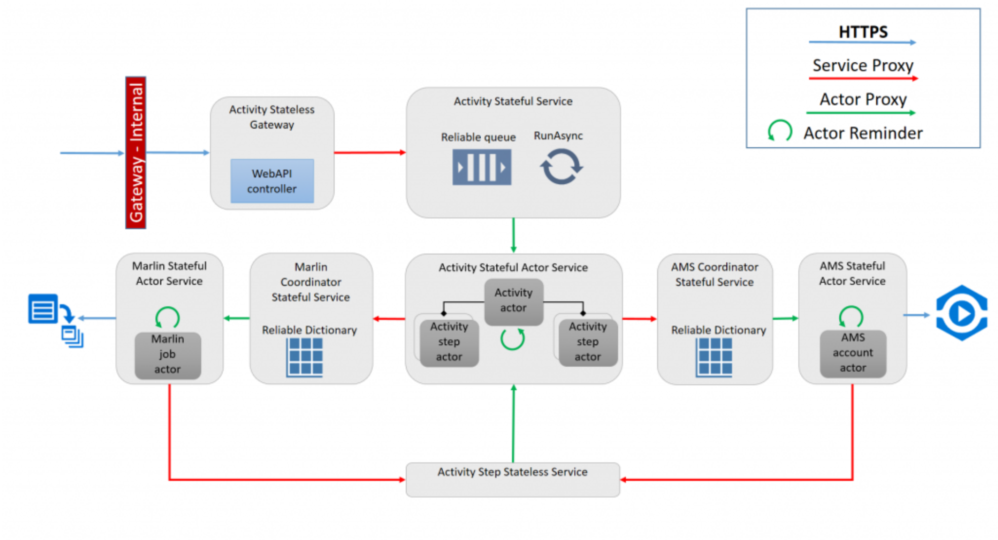
—Mike Goatly: Development Lead
Talk Talk TV



Relationship between the Service Fabric applications in the encoding and resource resolution application

Diving deeper into the activities application

Let’s now dive deep into the Activities application to see how this is composed of a variety of microservices each performing a specific role. The diagram below provides a detailed view of the Service Fabric microservices and the communication between them that make up the Activities application.


HTTPS
Service Proxy
Actor Proxy
Actor Reminder
Gateway - Internal
Activity Stateless Gateway
WebAPI controller
Activity Stateful Service
Reliable queue
RunAsync
Marlin Stateful Actor Service
Marlin job actor
Marlin Coordinator Stateful Service
Reliable Dictionary
Activity Stateful Actor Service
Activity actor
Activity step actor
Activity step actor
AMS Coordinator Stateful Service
Reliable Dictionary
AMS Stateful Actor Service
AMS account actor
Activity Step Stateless Service

The Activities Application and its microservices used for content encoding

The Activities application contains a number of microservices each of which performs a discrete role that encapsulate the logic required to create new content via Media Services and Batch, notify the back office systems and expose APIs. The functionality of each service is described below:

- **Activity stateless gateway service:** Provides a listener, based on Open Web Interface for .NET (OWIN), hosting an WebAPI controller that exposes the ability to create new encoding requests. This API is used for external interaction by services outside Service Fabric. It receives calls from the Azure Application Gateway service.
- **Activity stateful service:** Uses Service Fabric reliable queues for incoming work. This service then creates an Activity actor that represent the encoding work that needs to happen.
- **Activity Actor stateful service:** Represents a list of work items using a combination of .NET List<T> and Dictionary<T> in the actor state object to track the step execution and create the individual activity step actors. Being a stateful actor means no data is lost during failures.
- **Activity Step actors**: These are contained within the Activity Actor service and are responsible for coordinating the work steps via ServiceProxy and ActorProxy calls to other services. These services are used to encapsulate functionality and interactions with external systems, such as Azure Batch and Azure Media Services.
- **Marlin Coordinator and Marlin Actor Stateful Services:** These work together to schedule the Marlin encoding work and periodically check for its completion by the Batch service. Once completed, calls onto the Activity Step service.

- **Azure Media Service (AMS) Coordinator and AMS Actor Stateful Services:** These work together to schedule the AMS encoding work and periodically check for its completion by the Media service. Once completed, calls onto the Activity Step service.
- **Activity Step stateless service** exposes a Service Communication Listener, so that other Service Fabric services such as the Azure Media Services (AMS) Job Management services can interact with activity steps. It's like a call-back service—such as when work is completed by the Batch service and needs to wake up the ActivityStepActor to finish or update the work step.

Steps for encoding a new movie

Say TalkTalk wants to encode a new movie, let's walk through an example of how these microservices communicate with each other in order to achieve this.

- The TalkTalk admin website calls the *EncodeRequest*method, which is exposed using the Azure Application Gateway which forwards the request to the Activity Stateless Gateway service. Once this gateway service receives the request it uses a service proxy to communicate with the Activity Stateful Service.
- The Activity Stateful Service stores the encoding request on a reliable queue, *EncodeRequestQueueItem*. The service's RunAsync method periodically processes the queue and uses the actor proxy to create an Activity actor to represent the work.
- This activity actor in turn creates several Activity Step actors, which can transcode the movie into the various video formats required by XBox, PlayStation and other devices served from Azure Media Services. The Activity actor coordinates these activity step actors and ensures they execute in the correct order. To do this, a simple graph and is implemented on top of .NET collections.
- The Activity Step actor communicates with the AMS Coordinator Service, which implements a journal using Service Fabric Reliable Dictionary, this service in turn creates an actor to send a secure request over HTTPS to the Azure Media Services API. TalkTalk uses an actor reminder to periodically check whether the necessary encoding is complete. Once the asset has been encoded and the initial protection has been provided a call is made back to the Activity Step stateless service and the original Activity Step actor is notified.
- The Activity Step actor then contacts the Marlin Coordinator Service, which maintains state in an actor and makes the call to Azure Batch to perform the Marlin encryption and DRM the media. The required keys are retrieved securely from Key Vault. At this point the protected asset is moved to Azure storage, and a notification is sent to the TalkTalk back office that the file is ready in the STB format.
- Throughout this process a reminder on each actor is used to periodically save actor state in DocumentDB, this serves two purposes – firstly to provide ad-hoc querying capability, secondly it provides a protection and backup of the data outside of Service Fabric.

Summary

Like many businesses, TalkTalk began by moving first their applications to the cloud using a lift and shift approach to get quick results. but now are turning to microservices to provide greater agility and scale. By moving its content encoding and resource resolution service to Service Fabric hosted as microservices that communicate with each other over standard protocols with well-defined interfaces TalkTalk were able to make service deployments easier, support scale demands, became more flexible, and enable upgrades of individual services. This microservices design approach has ensured that TalkTalk can more quickly deliver features and content to their customers.