BROADCAST THE BRIDGE
Connecting IT to Broadcast

Empowering Cloud Through Microservices

Essential Guide

EG

ESSENTIAL GUIDES

Introduction

By Tony Orme, Editor at The Broadcast Bridge

Monolithic software designs are rapidly disappearing to the history books as a combination of virtualized computing and microservices are taking broadcast infrastructures to new levels. However, to truly leverage their value, then we must not only change our approach to infrastructure design, but also develop our mindsets.

Reliability is at the core of any broadcast facility as the massive demands we place on infrastructures to achieve 24/7 broadcasting will stress any system to its limits. Although we've been designing infrastructures with A-B redundancy for many years, the massive cost involved in duplicating every piece of core infrastructure is daunting. This further impacts on scalability as every single piece of equipment in every critical workflow must also be duplicated.

Microservices allow us to scale systems while at the same time maintaining high levels of reliability. The distributed nature of microservices not only delivers greatly improved reliability, but also abstracts away the hardware so the application can reside on any compliant datacenter anywhere in the world. It is this abstraction that demands a change in our mindsets as we must start thinking in terms of functionality and not application.

This may seem like a subtle and unimportant difference but thinking in terms of functionality allows us to free our minds of hardware dependent systems. It's fair to say that at some point the microservice must run on hardware, and it does, but the hardware does not have to be under the direct control of the broadcaster. Take for example a transcoder microservice. This may form part of the broadcaster's infrastructure, but it may also be a third-party offering supplied as a cloud service provider.

Instead of thinking in terms of ownership, the broadcaster is compelled to think about leasing a service, or function. Not only does this provide massive technical flexibility, but it also enhances the business as workflows can be expanded to directly meet the needs of the viewer. That is, when a program audience peaks for ad-hoc events, workflows can be expanded using pay-as-you-go methodologies.

The unification of microservices using APIs further increases their power for broadcasters. Using the same technology that internet browsers and web servers require, broadcasters can relatively easily control microservices within their domain. APIs may be different for each vendor microservice, but they follow a specified protocol that empowers collaboration and encourages dynamic scalability to deliver flexible workflows and solutions to meet the needs of today's viewers.

Security should be at the heart of any broadcaster's mindset and microservices have this built into their core. Although they are highly flexible and scalable, microservices maintain very high levels of security and help keep broadcasters high-value media assets safe.

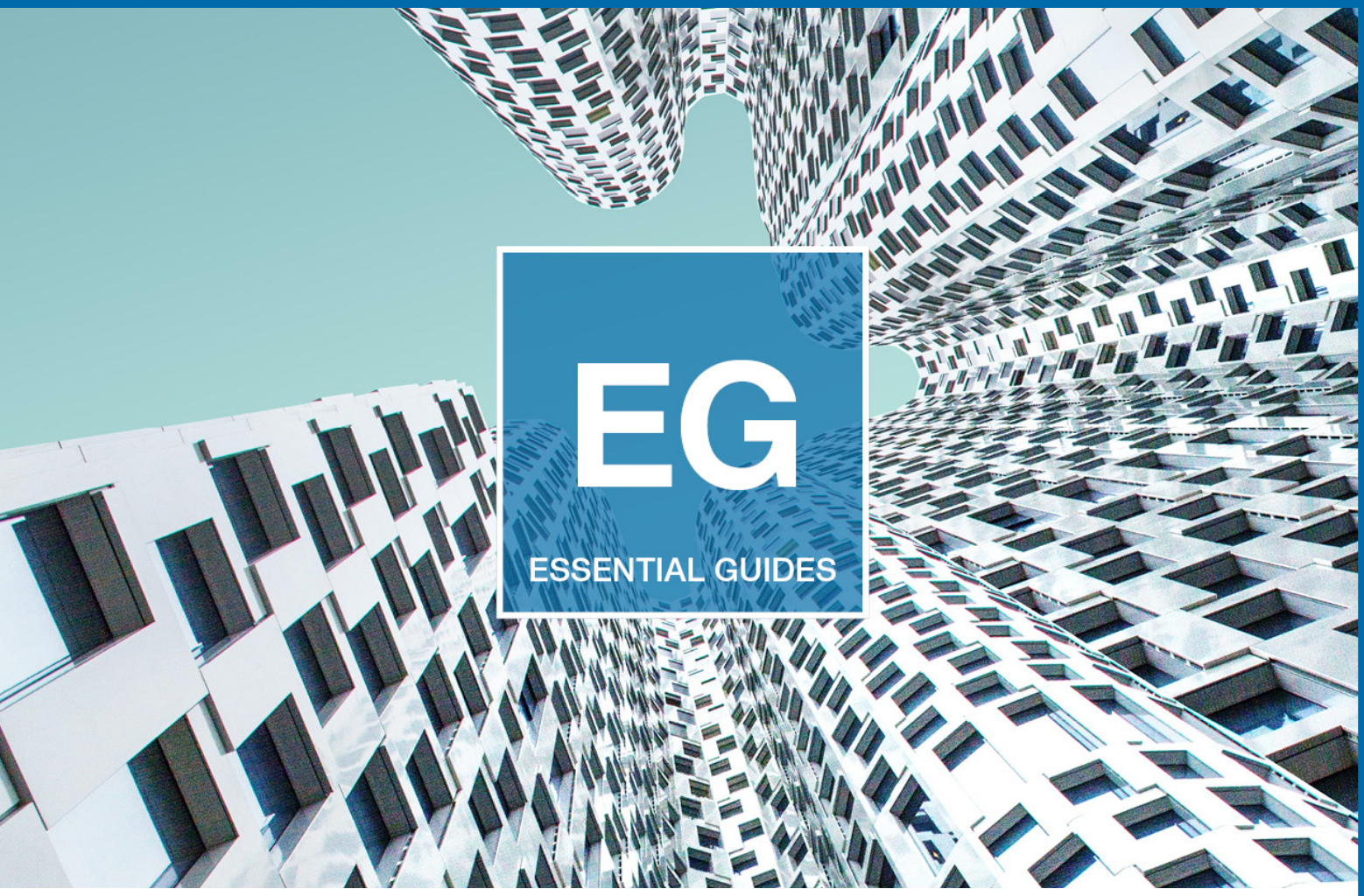
Distributed networks and infrastructures are further encouraging us to think in terms of mesh and not point-to-point. That is, a resource no longer should be considered in isolation, but instead, how it fits into a much bigger network where it can be dynamically allocated. This isn't just about thinking how a server can be allocated to perform a specific task, but how it's whole resource can be best utilized over multiple functions.

Tony Orme.

Microservices are starting to influence broadcasting both in the way we design infrastructure, and how we can dynamically use it. Broadcasters are no longer tied to static mindsets and can instead develop to take full advantage of scalability, flexibility, and resilience.

Tony Orme
Editor, The Broadcast Bridge

Empowering Cloud Through Microservices



By Tony Orme, Editor at The Broadcast Bridge

Simply moving workflows and software applications to virtualized infrastructures – whether public or on-prem – will not leverage the power of cloud. Monolithic software programs and static workflows all conspire against the broadcaster when they are reaching to achieve the flexibility, scalability, and resilience that cloud systems promise.

Broadcast infrastructures demand high availability and resilience, and scalability has always been an aspiration, but the peak nature of program productions has rendered this virtually impossible in traditional broadcast infrastructures.

If you build your infrastructure to handle the largest events and the busiest days with full redundancy, it sits idle most of the time. Cloud and virtualization not only deliver high availability and resilience but also massive flexibility and scalability. However, this is only possible if the infrastructure is built with cloud and virtualization in mind from the beginning.

Microservices both present a new method of designing and building software-based infrastructures and encourage a new way of thinking. A microservice isn't necessarily owned, but instead leased for the duration of the program or service it serves. This reflects a major change in how we think about making programs.

Instead of having to procure and find funds for capital expenditure that justifies the spend for years to come, we now have the opportunity to build entire broadcast infrastructures using pay-as-you-go methodologies.

Improving Reliability

Software is traditionally built using a monolithic design which means that a huge software release is provided for a single application or workflow. It is almost impossible to completely test the software prior to release due to the millions of combinations of inputs and outputs that are available. This can result in each release introducing bugs that have unintended consequences and unpredictable outcomes. Software engineers have tried to improve on this situation by introducing functional libraries and object-oriented code. The idea being that code could be reused, and if it had been in service for some time then it was assumed to be bug-free.

Embedded systems such as proc-amps and standards converters have used monolithic code for some time. Reliability is easier to achieve as the input and output data is better understood and easier to replicate due to working in a closed environment. Furthermore, vendors working in closed environments are often writing software for custom hardware, so they have much better control of how the product behaves.

The challenge we now face with modern broadcast workflows is that they operate on open architectures, that is, we use COTS hardware with their associated operating systems. This has both decreased the cost of the capital expenditure and greatly improved flexibility through the application of software functionality, but in doing so, has massively increased the potential for complexity. Furthermore, monolithic designs are not only difficult to maintain and upgrade, they do not lend themselves well to scalability. One reason for this is that monolithic architectures cannot easily duplicate themselves and coordinate user requests to multiple instances of the same application. COTS hardware has limited processing and as requirements increase, additional hardware needs to be purchased, integrated, and configured. This is a process that can take weeks or even months. Simply running the software on a remote server only moves the issue outside. A monolithic piece of software still needs to be installed, integrated, and configured regardless of whose server it is running on.

Microservices both solve the challenges of monolithic software and build on the advantages of COTS type infrastructures. One of the reasons COTS is so powerful is that hardware is much more readily available than it is with traditional closed broadcast systems.

It is worth remembering that the type of customized infrastructures broadcasters need is at the top end of the technology scale, in other words, it's relatively expensive. But the high-end servers, switches, and storage broadcasters require are the same type of technology that other industries such as finance and medical are also using, so it is much easier to procure and support.

We can also ride on the crest of the wave of innovation that these other industries provide and microservices are just one result of these advances.

Less Is More

One of the original design philosophies of UNIX architects Ken Thompson and Dennis Ritchie was to keep the code reusable and modular. Consequently, UNIX has a host of commands that allow the output of one program to be piped into the input of another program. By keeping the operating system programs relatively small, they became much easier to maintain and support.

Microservices are following this similar proven design philosophy. By keeping functionality well defined with contained input and output data values, they are much easier to maintain and support. Other benefits also include improved security and scalability.

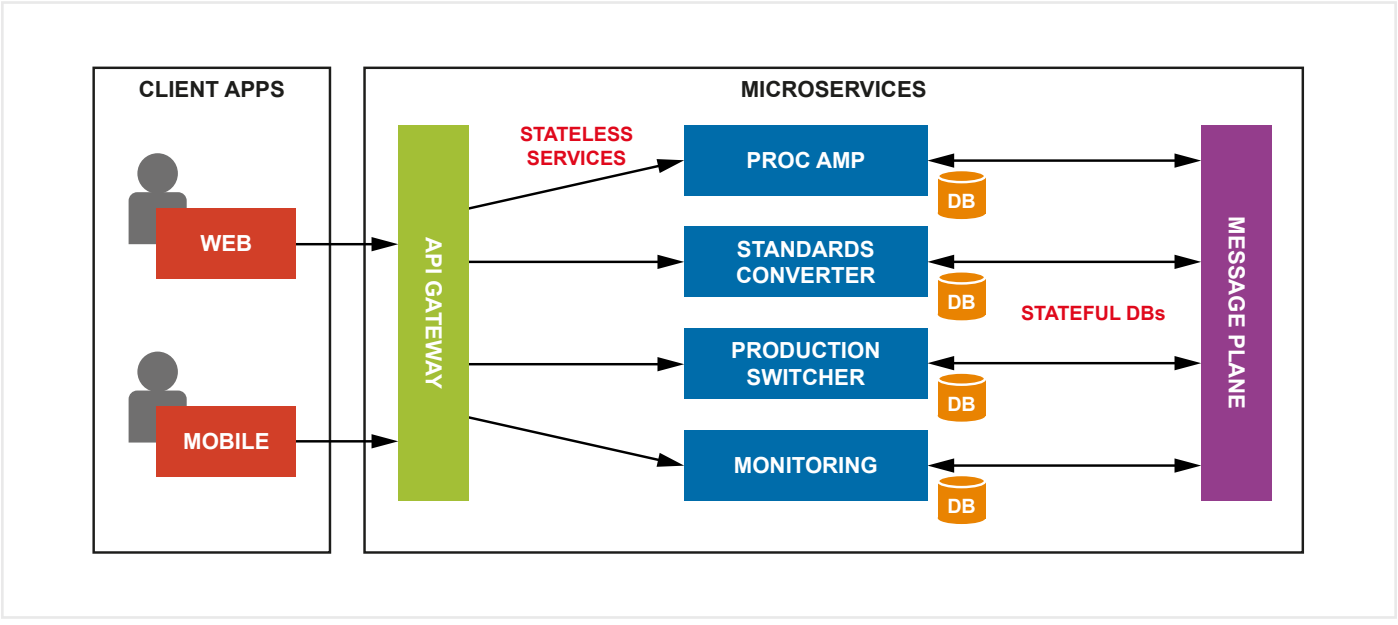


Figure 1 – Microservices are stateless allowing the API Gateway to schedule jobs within the workflow as requested. The user doesn't know, or need to know where the microservice physically resides, only that it is a service available to them.

In the same way that a house consists of thousands of bricks, all working together to make a huge building many times the size of one brick, microservices combine to deliver highly flexible, scalable, and resilient workflows that are much greater than the sum of the parts.

Key to understanding the microservice workflow is to appreciate the philosophy of building systems consisting of smaller parts. Unlike a brick house, we can pull the whole microservice workflow apart, delete it when it's not needed, and then reconstruct it again in a matter of minutes through the appropriate management software. Similar to the UNIX example above, microservices can be tested individually and can be easily installed, upgraded, or rolled back without impacting other microservices. This means that a small bug in one microservice will not bring down the whole system, improving the reliability of the system even when going through regular updates.

Broadcast facilities have worked with the modular mindset since the first television broadcasts, but these have always been fairly static. We have been able to design some flexibility and scalability into the infrastructures through assignable matrices and pluggable jackfields, but the reality is that we've been saddled with having to design for peak demand. No matter how flexible we try and make an infrastructure, the static nature of single functionality equipment has been a limiting factor. However, COTS infrastructures combined with manageable microservices are delivering untold levels of flexibility and scalability.

Transcending Cables

Servers, switches, and storage devices all need to be connected, but the dynamic nature of IP networks means that we don't have to constantly reconnect devices through circuit switched topologies. Instead, packet switched IP networks when combined with SDNs (Software Defined Networks) provide more flexibility and resilience than could ever be achieved with traditional broadcast systems by centralizing the control and management of network endpoints. When partnered with public cloud services this further reduces the need to design for peak demand.

To free ourselves from peak demand design mindsets we must have scalable infrastructures. Building a datacenter does not necessarily free us from this restriction. It's true that we may be able to take advantage of the statistical nature of the infrastructure needs as it's unlikely that every studio will be running at the same time. However, broadcasters are expected to, and often mandated to, provide the best possible coverage for major public information events such as the election of a country's leader. In such a case, the peak demand becomes a major issue.

Overcoming this limitation is achievable through a combination of providing scalable resource through public cloud services with on-prem data centers. The beauty of microservice type systems is that we are effectively abstracting the functionality away from the underlying hardware. From the perspective of the software, it doesn't matter whether the microservice is running in an on-prem data center or a public cloud as the platforms for both are similar and quite often the same. This is where the power of microservices manifests itself. A broadcaster can still design their on-prem data center for the average demand of their facility but plan their design so that additional scale and functionality can be provided in the public cloud on demand, such as in the case of a major public information event.

Public cloud services are delivering untold opportunities and possibilities for broadcasters, but they do not lend themselves well to static designs. Where they excel in terms of technology and cost is when being scaled in dynamic systems. However, some broadcasters do not always need this kind of flexibility as their average infrastructure requirement may not deviate too much from the average for most of the time. In these circumstances, having an on-prem datacenter is the best solution for their day-to-day needs. But when the major event happens, then they need to be able to scale quickly and effectively, and this is where the public cloud excels.

For broadcasters that have highly dynamic requirements, they would benefit from a purely cloud infrastructure. The great news is, as microservices can run equally well in on-prem datacenters as in the public cloud, the whole infrastructure becomes very scalable, certainly well within the requirements of many, if not all broadcasters.

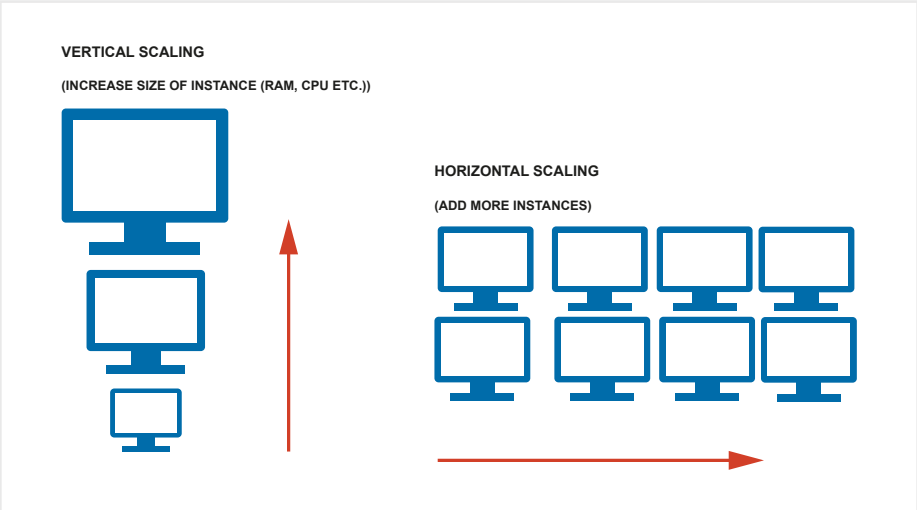


Figure 2 – Microservices can scale vertically and horizontally, or both simultaneously. That is, resources can be increased and allocated vertically, and simultaneously the number of services can be increased horizontally. The microservices are not tied to specific computer hardware and can scale across multiple devices.

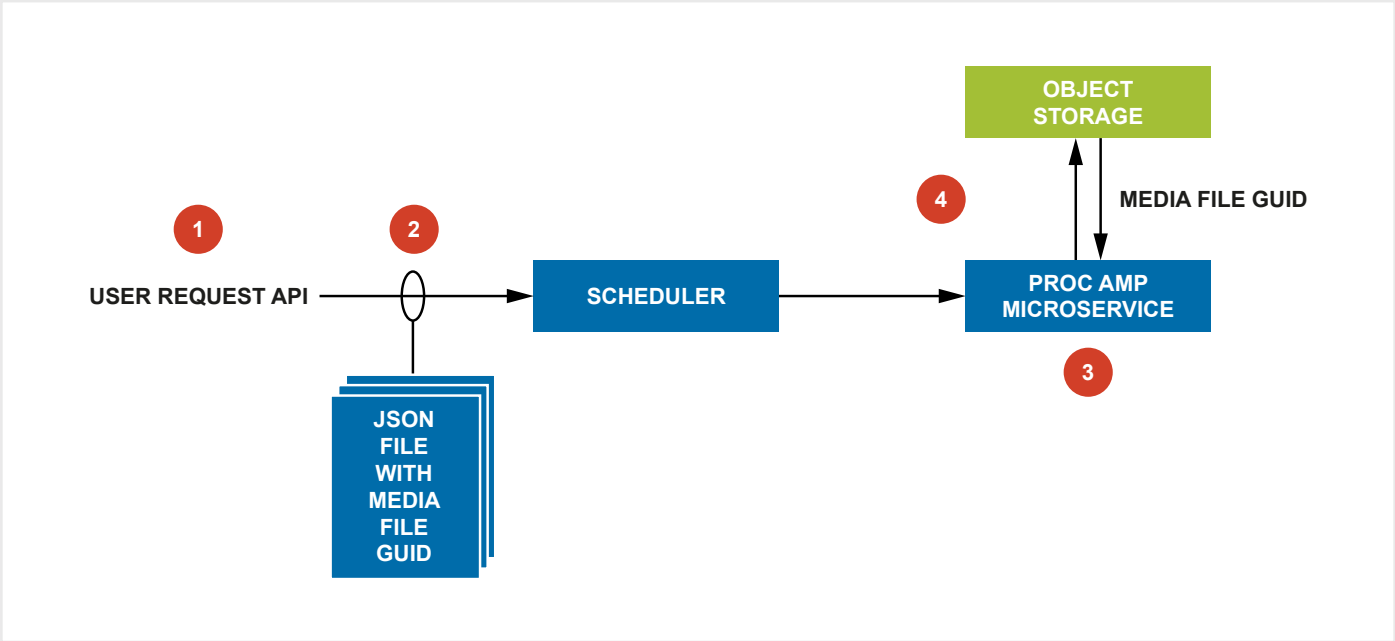


Figure 3 – The user initiates a job at (1) and sends an HTTPS request through the API with an attached JSON data file containing the media file GUID. (2) the scheduler decides which ProcAmp microservice to send the request to (there could be many depending on the workload). (3) the ProcAmp microservice accesses the media object storage using the GUID. (4) the object storage also provides the storage for the processed media file. When complete, the microservice sends a message to the scheduler which in turn responds to the user through the API referencing the processed media files GUID. The high-datarate media file is kept within the location of the high-speed object storage to improve efficiency and does not need to be streamed to the user.

Microservice Localization

When we speak of the public cloud it's worth remembering that the datacenters do physically exist. Admittedly they're often shrouded in an element of secrecy to maintain their security, but they are physical entities. This is important when planning the infrastructure as latency can be impacted by physical location, especially when considering human interface controls such as sound consoles and production switchers. Another reason for microservices that can run on hybrid infrastructures is to manage these timing issues.

As microservices are abstracted away from the hardware and use common interface control and data exchange, they can be moved between different datacenters. This provides a massive amount of customization as the microservices, with their container management systems, facilitate localization.

HTTPS And JSON

In part, microservices are flexible due to the control interface and data exchange methods they have adopted. They use the same technology that internet servers and web pages use, that is, HTTPS/TCP/IP and JSON data structures. Not only does this allow microservices to run on any cloud infrastructure, but it also opens the design to every software library and tool available used for web page development, further expanding the opportunities for broadcasters in terms of available and proven technology.

HTTPS (Hyper Text Transfer Protocol Secure) is used by web pages to exchange data with web servers. This protocol encourages a stateless method of operation and by doing so, the software application focuses on processing a chunk of data and returning the results to the requester. Stateless transactions don't need to reference earlier transactions to accomplish their task. Microservices are stateless further allowing them to take advantage of the sender-receiver model. That is, a software app sends an instruction to the microservice, it then processes the data and sends it back. This is the same method a web browser uses when it requests a web page from the server. But key to this is how the data is exchanged and represented.

HTTPS messages are an efficient method for sending a few hundred bytes of data. However, when the data becomes larger, as found with media files, HTTPS data exchange becomes extremely inefficient and potentially error prone. A much more reliable method is to provide a relatively small amount of data in the HTTPS message with information about where the media file is stored, and this is provided by a JSON file (JavaScript Object Notation). The massive media files are not embedded in the JSON data structure but instead it contains references to the storage locations of the media.

This opens outstanding flexibility as the microservice is not only abstracted away from the hardware but is further abstracted from the physical storage locations. Assuming object storage is used then the reference to the media file is a globally unique identifier (GUID), which means the microservice isn't restricted to a single server or domain.

For example, if a system needs the luminance gain increased in a media file, it will send an HTTPS message with a JSON file to the proc-amp microservice, the JSON file will contain GUIDs for the source and destination media file as well as the control parameters such as “increase luma gain by 3dB”, when the microservice has completed the operation, it will send a message back to the controller. And this is where the stateless part of the system comes into play, the controller issuing the proc-amp message does not know physically where the proc-amp microservice is, it can be on-prem or in a public cloud, and the proc-amp load balancer keeps a check of the number of proc-amp microservices it has available to it and then sends the message to the free proc-amp microservice.

Interface Simplicity For Own Build

Although the underlying process may sound complicated, and it is, the good news is that all this is taken care of by the system managing software, the containers that coordinate the microservices and their loads, and the microservices themselves. From the perspective of the user, the workflow is a series of HTTPS messages and JSON data files.

JSON files are incredibly powerful as they are human readable, so they can be edited with a text editor, and facilitate massive control over the microservice. It’s possible for an engineer or technologist to build their own workflows as the JSON structures are well defined by the microservice vendor and therefore easy to operate and daisy chain together.

Another interesting aspect of this philosophy is that of logging. There are two main reasons why we want to log microservices, first to understand how resource is being used and when, and second to provide data to facilitate forensic analysis should anything go wrong.

Optimizing workflows is incredibly important for scalable systems, especially when using pay-as-you-go costing models. Knowing when and why microservice allocation scales both up and down helps broadcasters make best use of their infrastructures.

Although datacenters and public cloud services are incredibly reliable, things do occasionally go wrong and having a forensic audit trail helps understand how to avoid issues in the future. Humans also make mistakes and understanding how and why this occurs helps prevent incidents in the future.

One of the challenges we have with logging is knowing what to log and when. If every message is logged for every microservice then the database will soon fill up, so a compromise must be found.

Conclusion

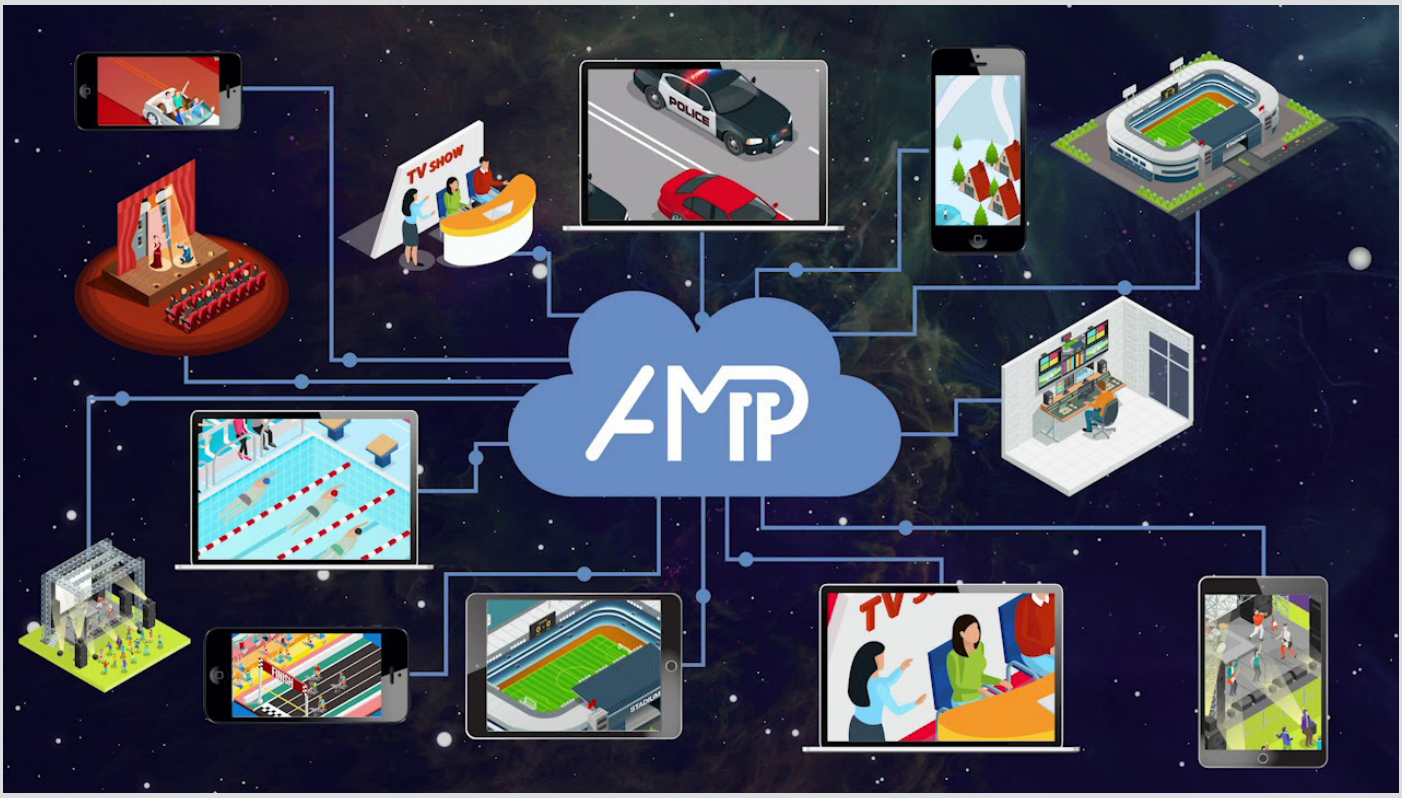
Microservices are empowering broadcasters to think differently about their workflows and deliver scalable, flexible, and resilient infrastructures. The complexity of their operation is abstracted from users allowing engineers to not only design systems but build their own adaptions to them when needed. When configured with the right mindset, microservices, on-prem datacenters and public cloud services can work harmoniously together to scale as needed to deliver truly flexible, scalable, and resilient workflows.

The Sponsors Perspective

The Answer Is: “Yes We Can!”

By Ian Fletcher and Chris Merrill

Flexible architecture opens new business possibilities.



Elevate Broadcast Pte Ltd. was one of the early adopters of Grass Valley’s agile media production and distribution platform, AMPP. Ever since their adoption of AMPP, they’ve been regularly using it for a wide variety of projects, from simple signal transport and monitoring to full live remote productions in the cloud. Elevate Broadcast has found that transitioning to a modular, microservices style architecture has increased their ability to quickly respond to the changing needs of their customers.

Dennis Breckenridge, CEO of Elevate Broadcast, explains: “One of the challenges in many of the IP environments is that you end up with all these gateway devices that are plumbed together to create a workflow. We can make that work for some situations, but it doesn’t have the same flexibility. You can’t say today I need eight inputs and tomorrow they need to be outputs. Or you can have a video switcher for this show but use those same resources to shuffle audio for the next show. AMPP doesn’t force you to make these decisions that you then have to live with.”

Supported by

While many of the projects using AMPP do have elements of cloud operations in them, Breckenridge was quick to point out that the flexibility that AMPP provides makes it just as useful in an on-prem environment.

“Last year we built out a big production center. In that case, AMPP is connected to a SMPTE ST 2110 world. The nodes sit in our data center. Then we use it for all kinds of things.

“We use AMPP extensively for format conversion from 1080i to 1080p productions, to do contribution to AI engines for editing or other processing. We use AMPP if we need to post process any signals, for example to multiplex or shuffle audio and then convert the feed to SRT or RIST. Rather than going out and buying converters and all those types of edge devices, we just feed the 2110 signal into AMPP, and it provides what we need.”

GV AMPP Architecture

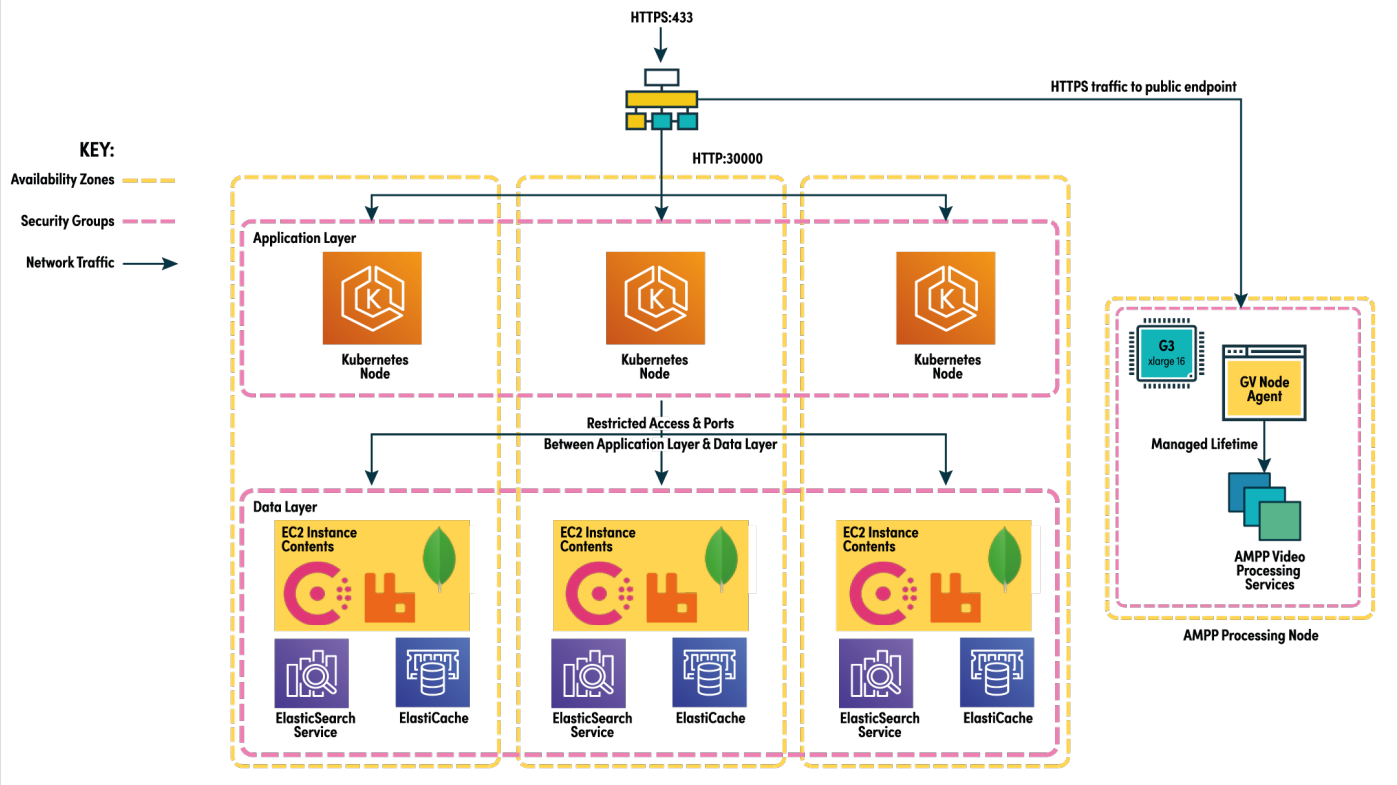
AMPP is a cloud-first microservices architecture that consists of a Grass Valley operated multi-tenant control plane – which is provided as SaaS – as well as a private customer video processing data plane that can either be in the cloud or on the ground. This enables extremely flexible workflows that have all the advantages of the cloud while recognizing that for some use cases, processing video at the edge makes more economic sense.

Cloud First

AMPP takes advantage of all the native services available in public cloud platforms. It consists of a set of microservices that are distributed across many physical computers in multiple availability zones. These architectures are normally defined as being high availability because the work is distributed across many microservices and one of which can fail without impacting the overall performance of the system. This provides better performance and reliability than you would get with a traditional “lift and shift” approach to the cloud which means taking some traditional monolithic software and simply running it on a dedicated VM in the cloud.

Microservices And Kubernetes

A single AMPP platform control plane is distributed across clusters of compute in different availability zones. AMPP operates on platforms distributed around the world so that customers access a platform that is local to them. Managing many microservices distributed over multiple data centers requires a management layer that handles the lifecycle of stopping and starting all the individual services and managing the resources they have available. Grass Valley uses Kubernetes to manage the control plane. Kubernetes groups containers that make up an application into logical units for easy management and discovery. Kubernetes builds upon 15 years of experience of running production workloads at Google, combined with best-of-breed ideas and practices from the community.



Single platform view - platform application and data layer spanning three availability zones.

Supported by

The AMPP Data Plane

Real-time video processing happens on the AMPP data plane. The data plane infrastructure may run on-premise or on a public cloud hosted virtual machine and is private to an individual customer account.

Many individual AMPP applications can be deployed on a single compute node. These apps can be stopped and started individually as needed, but they all share access to a common set of shared 10bit YUV uncompressed video flows so that multiple apps can interact with the same frames of video very efficiently without incurring any significant latency.

Within the same data plane, you can run many copies of the same app with its own specific configurations. These are called workloads and can be managed from a central application called the Resource Manager. The advantage of this approach is different productions can have their own workloads which can be stopped and started as a block while preserving all their individual show setup.

A New Way Of Working

While it is common to begin experimenting with AMPP as a one-to-one replacement for a specific hardware-based workflow, its true power lies in its ability rapidly provide whatever workflows are required at any given moment.

“The beauty of AMPP,” said Breckenridge “is that it is a toolbox that can be applied in so many ways. Before AMPP we had to build up a stock of converters and changeovers, clean quiet switches, and routing panels – a whole warehouse of purpose-built kit. Now we can be much more dynamic. We can add more inputs and outputs, we can scale the network, we can manage all different types of flows, and then bring all of that into whatever production environment we need: SDI, IP, cloud, hybrid… It really doesn’t matter.”

Ian Fletcher.

Chris Merrill.

BROADCAST
THE BRIDGE
Connecting IT to Broadcast

Themed Content Collection

EG
ESSENTIAL GUIDES

MEDIA

ARTICLES

For hundreds more high quality original articles and Essential Guides like this please visit:

thebroadcastbridge.com

12/2022

Supported by

Grass Valley
WE LOVE LIVE