Hybrid Integration
Thought Leadership White Paper

Cross-industry

Enterprise messaging in the cloud

A crucial element of a cloud infrastructure

Contents

Executive summary

Messaging is essential in a cloud environment, helping to provide resilience, performance and simplicity to IT systems. This is particularly true in systems that span diverse environments and in organizations where the various components may not always be available.

In general, there are two ways to consume messaging in the cloud:

- Deploy a software messaging system, such as IBM® MQ, into a cloud environment
- Use a managed messaging service, such as Message Hub, operated by a cloud provider

By deploying IBM MQ yourself, you must manage the messaging environment, but you get a proven, enterprise-grade solution built to your exact requirements. By using a managed service, the cloud provider operates the service, and you consume a more generic form of messaging. It's simpler and more cost-effective, but not tailored to your specific requirements.

Industry trends

One of the most significant changes over recent years has been the widespread acceptance of the cloud. Cloud deployments and cloud usage models are affecting all businesses whether they are moving some, all or none of their workloads to public cloud environments. IDC predicts worldwide public cloud services spending to reach USD 122.5 billion in 2017.[1]

Services are moving to the cloud, whether to a public, private or hybrid cloud infrastructure. Organizations are moving away from spending on infrastructure and focusing on creating more engaging digital experiences. Even the most experienced business leaders are continuously reassessing their hybrid cloud strategies in light of the quickly changing technology landscape. A robust, scalable, multitenant messaging system is a crucial element of any cloud infrastructure. The possibilities and opportunities in connecting things together in order to add business value are endless. A hybrid messaging system allows applications and services to communicate with each other asynchronously even in a multicloud platform.

Client needs and challenges

Background

Years ago, corporate networks often consisted of a mixture of networking protocols, including SNA, NetBIOS, and TCP/IP. Most applications used the network libraries directly, combining business logic and communications. A cross-platform, multi-protocol messaging solution such as IBM MQ was a major benefit for these organizations. Therefore, many organizations have existing messaging networks based on IBM MQ within and between their data centers and those of their business partners.

TCP/IP is used practically everywhere. Its flexible routing allows, in principle, any device to communicate directly to any other device. As networks became more dependable, synchronous communication using SOAP and then REST offered an expressive way to build distributed applications.

In a modern, cloud-based environment, messaging is no longer just about easily connecting disparate systems. It's now a question of choosing the most appropriate communication technologies for your overall architecture. Messaging offers higher cost efficiency and reliability than HTTP in connecting mobile and IoT devices.

Synchronous versus asynchronous communication

When designing a cloud-based system, you can choose various ways for the components to communicate. If you think of the system as a set of components making requests of each other, synchronous communication such as a REST API seems natural and is easier to understand. You might begin by waiting for the responses for all of the requests. But it depends on all of the communicating components being available at the same time.

As an alternative, the communication is seen as a flow of messages or events through the system. Often, requests won't have responses and, if they do, you don't wait for them before proceeding. That is known as an asynchronous view.

People often find synchronous communication easier to understand. Development tooling support for web services and REST APIs has simplified the building of distributed systems without having to know anything about networking. For many situations, this is entirely appropriate but sometimes a purely synchronous view of communication experiences performance issues.

For example, if you make a synchronous call to another service, what do you do if it's temporarily unavailable or performing extremely slowly? A complex cloud-based system that is entirely composed of synchronous calls is susceptible to performance and availability problems. Asynchronous messaging, as provided, for example, by IBM MQ, can be used to build highly available, responsive systems that can tolerate a degree of variability of performance.

In practice, a mixture of the two models is often appropriate, perhaps using synchronous communication within components and asynchronous communication between components.

Types of cloud environments

The following are key terms used throughout this description. Cloud environments make computing available as-a-service, hiding the inner workings from the users with the aim of making computing less expensive and more reliable.

The key characteristics of a cloud environment are:

- *Self-service*—Allows users to provision resources without human intervention, most likely using a web-based portal or an API.
- *Elastic scaling*—Enables scaling on demand, with higher degrees of automation.
- *Shared resources*—Offer economies of scale by using shared infrastructure and software, securely separating the resources at a logical level.
- *Metered usage*—Allows pay-as-you-go billing through monitoring, measurement and usage reporting.

There are five main cloud environment service models. Choosing the right one depends on how much control and visibility you need of the underlying resources. Table 1 describes each model:

Service model	Description	Examples
Infrastructure as a Service (IaaS)	This service manages low-level infrastructure, such as virtual machines (VMs) or bare-metal machines, virtual networks and virtual disks. You manage the contents of virtual machines, including operating system components such as firewalls.	• IBM Bluemix® Infrastructure • Amazon Web Services • Microsoft Azure • OpenStack
Containers as a Service (CaaS)	This service manages containers. You can use containers as part of an IaaS. However, a CaaS removes the need to manage anything other than your containers. Low-level components, such as firewalls, are managed by the CaaS provider.	• IBM Containers on Bluemix • Amazon Elastic Container Service • Microsoft Azure Container Service • Apache Mesos
Platform as a Service (PaaS)	This service manages only applications and basic scaling policies. It hosts application logic so that the application can be deployed into the cloud without having to manage the infrastructure.	• IBM Bluemix Cloud Foundry Runtimes • Amazon Elastic Beanstalk • Microsoft Azure App Service
Functions as a Service (FaaS)	This service manages only individual functions in a "serverless" environment. It provides hosting for small, event-driven pieces of application logic that can be highly scaled.	• IBM OpenWhisk • Amazon Lambda • Microsoft Azure Functions
Software as a Service (SaaS)	This service manages entire software applications. You simply configure as needed.	• IBM Cloudant® (NoSQL database) • Salesforce (CRM platform)

Table 1: Cloud environment service models, descriptions and examples

Many public clouds offer several of these service models within a larger platform, enabling you to pick and choose from an array of technologies for different parts of the solution.

For example, you might deploy IBM MQ in containers in your cloud's container service and run your client applications using the application run-times (PaaS). You might also use the platform's messaging service (PaaS) and its "serverless" environment for your event-driven applications (FaaS).

Many large corporations have their own clouds and use cloud principles within their data centers. Open-source tools such as Docker, Kubernetes and OpenStack can be deployed to create private clouds.

Depending on where the cloud environment is located and who operates it, the following terms are used:

- *Public*—Shared infrastructure, multitenant environment, managed by a cloud provider in their data center, for example, IBM Bluemix
- *Dedicated*—Separate infrastructure, single-tenant environment, managed by a cloud provider in their data center, for example, IBM Bluemix Dedicated
- *Local*—Separate infrastructure, single-tenant environment, managed by a cloud provider in your data center, for example, IBM Bluemix Local
- *Private*—Separate infrastructure, managed by your staff for your corporation in your data center
- Public, dedicated and local clouds are all examples of *managed clouds*, that is, clouds managed by a cloud provider.

Messaging use cases

The key concept in the following use cases is that the sender and receiver of a message do not need to be running at the same time or the same rate. This is known as temporal decoupling. Common use cases for messaging that involve temporal decoupling in the cloud include:

- Offloading time-consuming processing
- Smoothing variations in system load
- Processing event-driven data
- Integrating resiliency into IT systems

Offloading time-consuming processing

Imagine you develop a web application that requires time-consuming cloud service processing. You need to keep the web application responsive.

One way to handle this is to make the communication with the time-consuming processing asynchronous. The web application sends a message to request the processing and then continues without waiting. In the meantime, one or more workers receive the messages from the web application and perform the processing. Of course, the web application needs to be designed with this separation in mind, but it can remain responsive for the users no matter how long the processing takes.

Smoothing variations in system load

Parts of a system can become overloaded or degrade performance as the load increases past a certain point. Messaging can help in this case too. Techniques such as tuning the number of a pool of workers to the optimum size and using a queue of requests to deliver work to the workers can help redistribute the processing load.

Processing event-driven data

When an event occurs, one or more actions can be performed. This is usually achieved using publish/subscribe messaging. Subscribers indicate their interest in events by making a subscription with a messaging server or broker. When an event occurs, the publisher sends a message to the messaging server. The server examines the list of subscriptions and forwards the message to matching subscribers. The publishers and subscribers do not communicate directly.

This technique is beneficial because it can more easily adapt to changing requirements. For example, if you want to start logging requests for auditing or for performing some analysis, you simply add a new subscriber to an existing system. The producers of the requests don't need to be made aware of a change to the system.

Cloud services such as Salesforce and GitHub expose event-based interfaces, emitting events that you can consume in your applications.

Integrating resiliency into IT systems

A key principle in any architecture is that failures can occur. Components should be resilient to those failures. The same is true in cloud computing architectures. In a large, complex system, there are many components with different availability characteristics and maintenance schedules. If the system fails whenever any component is temporarily unavailable, it has a dramatic impact on overall system availability.

One technique for managing this is to use messaging to communicate between the components. In this way, when a component fails, you simply build a queue of messages for it to process when it becomes available or you choose to reroute those messages.

This is particularly useful when integrating with a cloud service, or communicating between cloud providers. You're not in control of all system components. You cannot control, or guarantee the availability of, another company's service. You also can't run a redundant copy of it.

A pragmatic way to handle this is to use a queue to build requests for the external service. Then, you can have a worker task that consumes the messages from the queue and performs the calls to the service. Messaging can help keep your systems resilient to unplanned outages and routine maintenance windows you might encounter.

How to choose a cloud messaging technology

Consider the following factors when choosing messaging technology for your cloud application:

- *Management costs*—How much management are you able to perform? Are you looking for a managed service or a solution you deploy and manage yourself?
- *Message delivery assurance*—Is it acceptable to receive messages more than once, or not at all? Do you rely on transactions to coordinate message delivery with other components such as databases?
- *Application complexity*—How much complexity are you prepared to manage in your application layer, versus offloading to the messaging layer?
- *Interoperability with existing components*—Do you already have software using messaging with which you need to communicate? Do you need applications running with different cloud providers to exchange messages?

How to deploy messaging in the cloud

As previously discussed, there are generally two ways to consume messaging in the cloud:

- Deploy a software messaging system, such as IBM MQ into a cloud.
- Use a managed message service provided as part of a cloud, such as IBM Message Hub.

Deploying IBM MQ in the cloud

You can deploy IBM MQ into a managed cloud or a private cloud provided by your IT team. If you use multiple cloud environments, you might choose to use IBM MQ deployed in all of the clouds to provide a common messaging backbone, rather than relying on different services in each cloud environment. This also provides performance and reliability benefits, as each application can communicate with low latency to a local IBM MQ server. Then, IBM MQ can reliably manage the exchange of messages across higher latency inter-data center links.

Deploying your own IBM MQ system gives you control over its design and capabilities, such as ensuring the security configuration matches your organization's requirements. However, by deploying IBM MQ yourself, you also manage the environment. You get much of the control that you get running IBM MQ in a traditional data center, but also a lot of the same requirements for monitoring, security and more. If you use an internal cloud, you may have more control over the hardware and software that underpin it, which you usually won't have in public cloud environments.

One of the features that distinguishes IBM MQ from managed messaging services is its support for transactions. Exactly-once delivery is greatly simplified by transactions, freeing the application from much of the complexity of reliable message delivery.

Using a managed messaging service

Most public cloud providers have one or more messaging services available, such as the IBM Message Hub in Bluemix and the Microsoft Azure Service Bus. As a managed service, the cloud provider takes on responsibility for deploying and running the service, and you simply use it.

To use a managed messaging service, it's simply a matter of provisioning the resources you need from your cloud provider. For example, if your application design requires publish/subscribe messaging for a particular type of event, you create an instance of the messaging service and configure a topic. Applications deployed on the cloud platform can be bound to the messaging service.

A managed messaging service offers one of the tightest integrations with the other services in that cloud platform. For example, the integration of the messaging service with the platform's event-driven application run time (FaaS) is provided and you configure it. This is the case with the integration of OpenWhisk with Message Hub in IBM Bluemix. You simply configure OpenWhisk with the name of a Message Hub topic to trigger OpenWhisk actions when messages arrive.

Cloud messaging services typically place a high value on scalability and availability, but a lower value on the assurance of exactly-once message delivery. Cloud messaging services are generally composed of clusters of redundant servers that collectively provide the messaging service. The messaging service might even be distributed across a wide area without you being aware, and information will usually be replicated so that it's not lost in the event of a failure.

In this environment, it can be expensive to provide exactly-once message delivery because of the technology choices that give excellent availability and scalability. In cloud messaging services, it's much more common to have at-least-once message delivery, meaning that there's a small chance that a message might be delivered more than once.

Managed messaging services do not offer general-purpose transactions as you might be used to in the data center. If you absolutely need exactly-once delivery from a managed messaging service, you generally have to write logic into the application. This logic either detects and discards duplicates or carefully tracks which messages have already been processed.

Technical considerations for running IBM MQ in the cloud

Although it's possible to "lift-and-shift" a complete IBM MQ infrastructure from a data center into a cloud, along with much of the application and infrastructure layers around it, it's possible that a design that served you well for many years in your data center can prevent you from using many of the benefits that you might expect when moving to a cloud.

For example, many IBM MQ topologies originated before dynamic scaling, high availability and self-service were a reality. If those characteristics aren't currently built into your solution, the act of moving to a cloud is an ideal opportunity to revisit such design decisions.

This may be as simple as opting to change from a manual method of provisioning IBM MQ servers to an automated provisioning mechanism. Or it may mean a change from a highly distributed IBM MQ server topology to a centralized hub of IBM MQ servers, providing the messaging capability for a wide range of applications.

Many of the technical considerations for running IBM MQ in the cloud are the same as those when running in other environments. However, there are a few major points that need special consideration:

- Persistent storage
- Security
- Cloud ingress and egress
- Scalability

Persistent storage

IBM MQ is known for its reliability. The foundation of this is reliable disk storage. Taking this reliability into the cloud means that you need to ensure that you have disk storage where disk writes aren't cached. For higher availability, you should consider whether you can replicate data across failure domains (also known as availability zones). Some cloud providers offer replicated storage (for example, Amazon Elastic File Storage). In others, you might need to manage it yourself with technologies like IBM Spectrum Scale™, Ceph or Gluster.

The two main types of storage to consider are networked file systems and block storage. Networked file systems are used for the IBM MQ multi-instance queue manager support, where two instances of the same IBM MQ queue manager have access to the same storage in an active-passive configuration, with automatic takeover in the event of a failure.

In contrast, block storage is attached to a single instance of IBM MQ queue manager at a time. This can still be used for a level of high availability in the cloud. You can dynamically deploy a new virtual machine or container to replace a failed one, using the same storage. You just make sure that your cloud provider offers an appropriate guarantee that only one queue manager instance accesses the storage at a time. This provides an effective alternative to failover-based high availability.

Security

The shared nature of cloud environments makes security a key concern – it becomes increasingly unwise to secure only the edge of the network. All data in motion and at rest should be secured: you should ensure that all IBM MQ channels are configured with TLS, use channel authentication for access control, and configure disk encryption either at the OS- or cloud provider-level. IBM MQ Advanced Message Security is also available to provide additional security protections, such as per-message encryption.

Cloud ingress and egress

A traditional data center network architecture usually includes a demilitarized zone (DMZ). This is used for internet-facing services, and provides an extra line of defense in the network. This pattern can be replicated with a software-defined network in a cloud environment. The major difference is that you'll probably need to rely on software-defined separation, rather than on physical separation of DMZ and private resources.

When connecting from the data center to a managed cloud, it's unusual to permit network connections from the cloud directly into the data center. For a dedicated or local managed cloud, it's common to use a virtual private network (VPN) to connect between the cloud and the data center. For a public managed cloud, a gateway component such as the IBM Bluemix Secure Gateway enables a secure tunnel to be established from the data center into the cloud that can then be used to carry IBM MQ connections.

You can also use a feature of IBM MQ that allows the connection to be established by the message receiver, even though it doesn't have a public IP address.

Scalability

When considering scalability, IBM MQ client applications should probably be scaled separately from IBM MQ servers, which means running them in separate virtual machines or containers. This means that IBM MQ cloud topologies would typically use IBM MQ client connections to communicate between applications and queue managers.

Scaling IBM MQ up is straightforward, using either an IBM MQ cluster of queue managers, or a load-balanced set of identical queue managers, depending on what messaging patterns you use. Scaling down is more complex because it's good practice to remove a queue manager in a controlled manner to be sure all messages are safely processed.

The other important consideration with scalability is message ordering. It's normal in the cloud to scale out to multiple servers, rather than scale up to use a bigger server. This brings with it the possibility of concurrent processing, which means messages might be received out of sequence. IBM MQ provides features to allow groups of messages to be handled in small ordered batches. You can also choose to manage this yourself in your application.

Hybrid cloud messaging scenarios

Most businesses using cloud computing use it as part of their overall IT environment. There are many scenarios in which messaging can be used to communicate between cloud systems. We describe these as hybrid cloud messaging.

Cloud applications connecting to IBM MQ in the data center

In this scenario, applications are deployed in the cloud but they don't use a cloud messaging service and IBM MQ isn't deployed in the cloud. Instead, the applications make client connections back to the enterprise IBM MQ backbone, as shown in Figure 1.

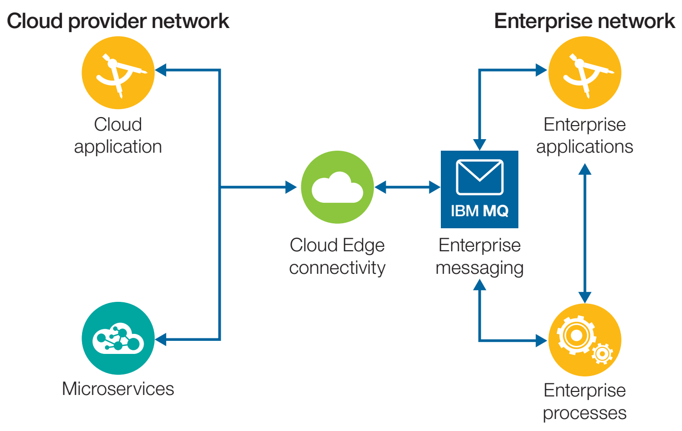

Figure 1: Cloud applications connecting to IBM MQ in the data center

This scenario can be used with managed or private clouds. With a managed cloud, it's typically only used when there's a relatively modest investment in cloud, perhaps using unmodified applications that have been moved from the data center into the cloud.

A difficulty with this scenario is that the latency from an off-premises managed cloud to the data center is much higher than the latency for connections within a data center. Depending on the design of the application, this might lead to performance problems.

Cloud applications connecting to IBM MQ deployed in the cloud

In this scenario, IBM MQ is deployed into the cloud along with the applications. The applications make client connections to IBM MQ in the cloud. If an off-premises managed cloud is used, it's likely that connections to IBM MQ are made over the public internet. So it's still necessary to be careful about securing the connections. (see Figure 2.)

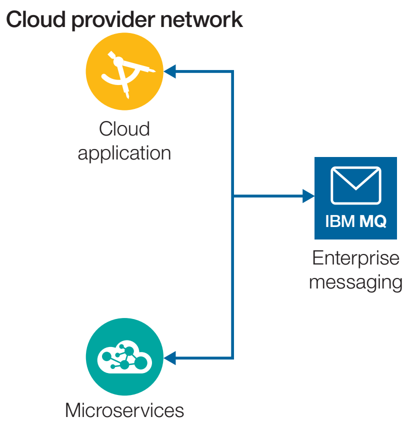

Figure 2: Cloud applications connecting to IBM MQ deployed in the cloud

Bridging the enterprise IBM MQ backbone into the cloud

In this scenario, IBM MQ is deployed into the cloud as an extension of the enterprise messaging backbone, rather than an isolated network. Routes are established between the enterprise messaging backbone and the IBM MQ systems running in the cloud so messages can be exchanged between the environments. Best practices for IBM MQ topologies spanning networks can be employed, as shown in Figure 3.

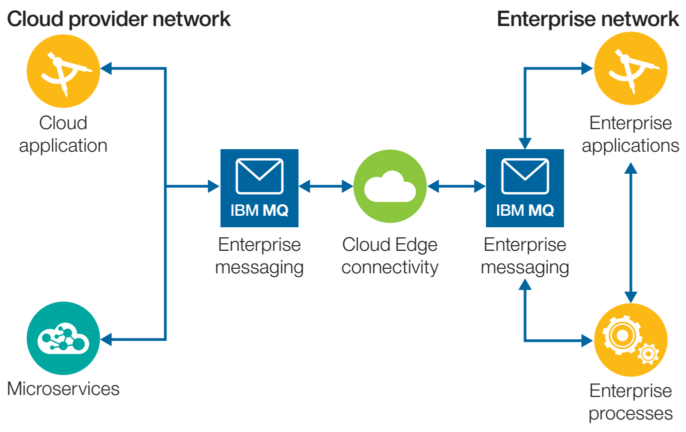

Figure 3: Bridging the enterprise IBM MQ backbone into the cloud

You can send messages from a public to a private cloud, even if the private cloud isn't internet-facing, by configuring the private IBM MQ queue manager to initiate the connection, and then flow messages from the public to private cloud.

Cloud applications connecting to cloud messaging services

In this scenario, the applications are deployed into the cloud, but make direct use of the messaging services provided natively in the cloud. It's likely that the programming interfaces for the cloud messaging differ significantly from IBM MQ. So the applications might either be new or adapted from earlier applications, as shown in Figure 4.

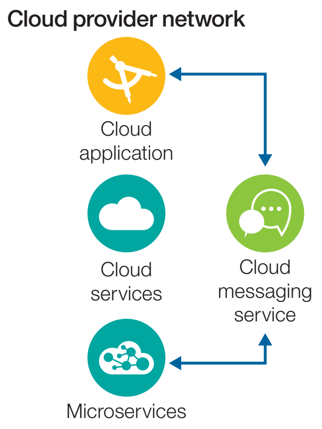

Figure 4: Cloud applications connecting to cloud messaging services

This scenario is most likely used when the cloud is being used as a platform for new applications. But there is no need to be compatible with earlier systems and the messaging capabilities provided by the platform satisfy the application requirements.

Bridging cloud messaging service to the enterprise IBM MQ backbone

In this scenario, cloud applications are making use of a cloud messaging service but it's necessary to exchange messages between this service and the enterprise IBM MQ backbone. Because two different messaging systems are being used, some kind of bridge is needed to join them. This bridge might be provided as a service by the cloud provider or a specialized application either deployed on the cloud or in the data center, as shown in Figure 5.

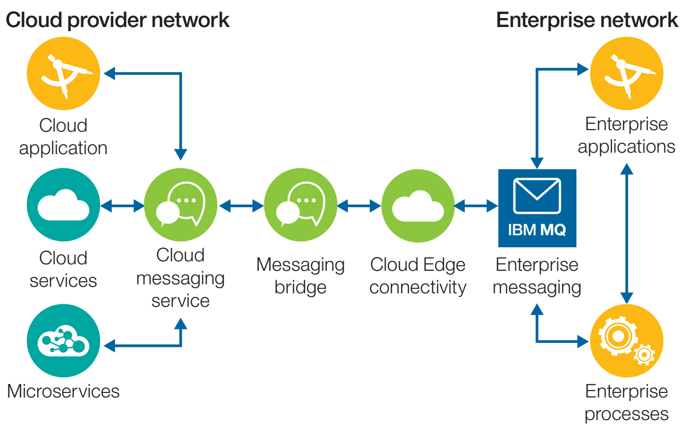

Figure 5: Bridging cloud messaging service to the enterprise IBM MQ backbone

Event streaming from a service in the cloud

In this scenario, a cloud service that can generate events is used. For example, there might be events generated by a cloud CRM system that would be of interest to applications in the data center. A specialized application in the data center subscribes to the events from the cloud system and transfers them into the enterprise IBM MQ backbone.

IBM solution and capabilities

IBM MQ has set the industry standard for throughput:

- 98 million messages per second on native InfiniBand and shared memory
- 75 million messages per second on 10-Gigabit Ethernet
- Industry-leading latency:
  - 770 nanoseconds for shared memory
  - 2 microseconds over native InfiniBand
  - 3.6 microseconds over Ethernet (10 GbE)

Conclusion

Messaging has an important part to play in building responsive, resilient, cloud-based IT systems. For applications running in IBM Bluemix, the IBM Message Hub service offers a fast and highly scalable messaging service, which is managed for you. For applications running in other clouds, or where you need more advanced enterprise features or seamless integration with an existing IBM MQ network, you can use IBM MQ.

For more information

To learn more about Enterprise messaging in the cloud, please contact your IBM representative or IBM Business Partner, or see: **ibm.com**/software/products/en/ibm-mg

About the author

Arthur Barr is a senior software engineer, working on all things related to IBM MQ in the cloud.

Andrew Schofield is chief architect for Hybrid Cloud Messaging. He has more than 25 years of experience in messaging middleware and the Internet of Things, with particular expertise in the areas of data integrity, transactions, high availability and performance. He works at the Hursley Park laboratory in England.

1 *Worldwide Public Cloud Services Spending Forecast to Reach $122.5 Billion in 2017*, IDC, February 2017