ukcloud

Pure commitment.

KEY CHARACTERISTICS OF CLOUD APPLICATIONS

version 2.0

CONTENTS

OVERVIEW

Although many legacy and enterprise applications can run in the cloud, only applications that are designed specifically for the cloud can truly capitalise on the benefits offered by cloud platforms. These include accelerated time-to-value, unparalleled elasticity and reduced costs.

This Blueprint explores the features of cloud applications that enable them to deliver these and other benefits, and explains why they're rarely found in traditional application architectures.

LEGACY, ENTERPRISE AND CLOUD APPLICATIONS

Many applications work in the cloud, but only some are designed for it. This is an important distinction because many organisations are attracted to cloud for the unique benefits it offers compared to traditional managed hosting platforms, including:

- Accelerated time-to-value — resources can be provisioned on demand to remove delays
- Unparalleled elasticity — additional resources can be allocated and released as required
- Reduced costs — environments can be continuously right-sized so that you only pay for the resources you need, when you need them

These benefits are most likely to be realised when organisations run applications that dynamically and automatically control the underlying environment. Otherwise, the size of the cloud environment remains relatively static, which means organisations don't benefit from cloud's elasticity, and can't realise the cost savings associated with releasing cloud resources when they're not required.

At UKCloud, we tend to divide applications into three broad types:

- Legacy applications. Typically, bespoke applications, often large and monolithic, that have been developed by or for the organisation, and use legacy application architectures such as mainframe, client/server or similar.
- Enterprise applications. Commercial-off-the-shelf (COTS) applications such as SAP, Oracle e-Business Suite, Microsoft SharePoint. These applications are typically designed for traditional physical or virtual environments and, like legacy applications, rely on large servers (scale-up) and complex hardware clusters for high availability.
- Cloud applications. Designed to leverage the agility and elasticity inherent in cloud platforms. They are available both as commercial Software-as-a-Service (SaaS) and bespoke applications, and are commonly developed using modern application frameworks and data services. Unlike legacy and enterprise applications, cloud applications are designed to scale out horizontally and to expect and tolerate component failures.

	Legacy applications	Enterprise applications	Cloud applications
Type	Bespoke	Commercial-off-the-shelf (COTS)	Bespoke or Software-as-a-Service (SaaS)
Infrastructure	Mainframe	Intel x86/x64 — virtual and physical	Intel x86/x64 — cloud infrastructure
Scale	Vertical – fewer, larger servers	Vertical — fewer larger servers	Horizontal — many smaller servers
High availability	Specialist fault-tolerant reliable hardware	Hardware failover clustering solutions	Resilient commodity hardware, application layer fault tolerance
Disaster recovery	Active/passive — single site which fails over to recovery site	Active/passive — single site which fails over to recovery site	Active/active – multiple sites in use at all times
Security	Secure physical environments and restricted mobility	Secure physical environments and restricted mobility	Secure data assets embracing mobile end-user devices

Table 1. Legacy, enterprise and cloud applications compared

So it is easy to see that although legacy and enterprise applications can run in the cloud, only cloud applications can capitalise on the unique benefits of cloud platforms.

This Blueprint explores the features of cloud applications that provide the levels of availability, security and elasticity that are rarely found in traditional application architectures.

CHARACTERISTICS OF CLOUD APPLICATIONS

Cloud platforms offer real benefits in terms of improved agility and reduced costs, as long as applications can take advantage of the elastic capabilities provided by the platform. To enable cloud applications to operate securely, resiliently and elastically in the cloud, they need specific architecture characteristics.

Automated provisioning

A key characteristic of cloud is the ability for the entire cloud-based solution — including the infrastructure — to be represented as code, so that the process of deploying, testing and releasing software can be fully integrated.

Cloud applications should therefore be developed using tools such as continuous integration and continuous deployment (for example, Jenkins, Chef, Ansible), and automated testing (for example, Cucumber). These tools commonly provide native support for our Cloud Native Infrastructure (powered by OpenStack) and so make it easy to implement an end-to-end continuous integration solution. UKCloud Compute-as-a-Service also supports features such as server templates and fenced clones, which further facilitate the provisioning and testing of cloud applications.

Ideally it should be possible to automatically provision the entire application architecture as required. Because many legacy and enterprise applications require manual provisioning and customisation, it's very difficult to re-provision them, which places more emphasis on backup and recovery as the applications will need to be restored from backup if problems occur.

Cloud applications, on the other hand, are designed so that they can be quickly and easily re-provisioned to a 'known good' state. This is most easily achieved if the entire application stack (application, middleware, operating systems, virtual machines, networks etc) can be automatically provisioned. This is sometimes referred to as 'infrastructure-as-code' — meaning that the infrastructure components such as firewalls, networks and virtual machines can be dynamically provisioned and reconfigured via code rather than manually via a GUI or portal.

See http://devops.com/blogs/meet-infrastructure-code/ for more information on this topic.

USING A CONTINUOUS INTEGRATION TOOL

A continuous integration tool such as Chef can be used to automatically:

- *Pull source code from a central repository such as Git*
- *Provision virtual machines on the UKCloud Cloud Native Infrastructure*
- *Deploy the application*
- *Perform automated integration testing*
- *Promote the release into production*
- *Tear down the non-production environment when finished*

Resilience and redundancy

When designing a cloud application, it's important to consider how it will handle error conditions. Error conditions can occur for a number of reasons, including:

- Human action — operators might make an error (misconfiguration) or cause a component to become unavailable, eg by taking it offline for maintenance.

- Infrastructure issues — hardware devices, networks and entire sites can fail. Most infrastructure components are deployed in a resilient configuration and recovery times can range from near instantaneous to a number of hours.

- Software issues — software and web services can fail for a variety of reasons ranging from memory leaks and thread-safe issues through to poorly tested changes. As applications increasingly rely on third-party-controlled web services controlled (eg postcode lookups, payment authorisation), organisations cannot assume they will have visibility of all changes and issues which might have a direct or indirect impact on their application.

Organisations should therefore expect failures to occur, and develop their applications to respond gracefully to failure conditions. For example, customers should consider how their application will handle:

- Transient network failures — what if the application client can’t connect to the application server?

- Component unavailability — what if an application component is unavailable? What if the application can’t connect to its database?

- Unexpected configurations — what if the response generated by a component is not as expected? For example, a postcode lookup request to a web service that returns an unexpected value (such as an error code).

Traditionally, legacy applications and enterprise applications differentiate between high availability (keeping the system running when business-as-usual failures occur) and disaster recovery (restoring application service in the event of a catastrophic failure). In the cloud, these two failure conditions can be treated as one, because applications can be designed to expect and tolerate catastrophic failures, such as the loss of a data centre. However, for clarity, the following section describes how organisations can address high availability and disaster recovery concerns with their cloud applications.

High availability

When designing a cloud application, organisations need to consider how high availability will be achieved. In traditional infrastructures, specialist hardware is commonly used to provide high degrees of fault tolerance and reliability. As described above, it should be expected that individual components will fail (for example, system failure, misconfiguration, maintenance activity). UKCloud therefore focuses more on resilience than reliability to ensure that the cloud service continues to operate regardless of individual component failures. Cloud applications should be architected to expect failures to occur, including the implementation of sufficient error checking and redundancy to ensure that requests can be serviced by alternative means when required.

For example, when an application accesses a database service, it should:

1. Check that the database service is running

2. Respond gracefully if the primary database service isn't running by, for example:

    - Queuing the request so that it is processed when the database service becomes available again; or

    - Routing the request to a secondary database service, one that is perhaps running on another node, at another site or in another zone, or even on another cloud platform

    - Whereas legacy and enterprise applications would simply fail if the database service was unavailable, cloud applications can be designed to expect the failure and continue running in one way or another.

    - **Disaster recovery**

    - Legacy and enterprise applications typically implement an active/passive solution for disaster recovery: the application runs actively in one data centre and operates in standby in another data centre. The standby service could be anything from a real-time replica of the production service (for example, using synchronous replication) to a cold standby that involves restoring the application from a backup. In the event of a disaster at the primary data centre, the time taken to restore the application service (RTO) could range from seconds to days, and the amount of data potentially lost (RPO) could range from none to several days' worth.

    - In the cloud, all sites (or zones) are active and capable of running active workloads: applications use this capability to offer zero downtime and zero data loss. To this end, applications should be designed to expect an entire site to become unavailable, and to gracefully resume operations within the remaining site until the other site becomes available again. This can be achieved using principles such as micro-services and loose coupling (both explained below) to enable the techniques for delivering high availability to be extended to cater for more catastrophic failures.

There are two primary challenges when running an application actively across two sites:

- How users are directed to each of the two sites. This can be solved by using solutions such as global load balancers (such as F5 Networks' Global Traffic Manager), content delivery networks (Akamai) or DNS-based solutions (for example Neustar UltraDNS).

- How data consistency is handled so that transactions can occur at either site. This is often achieved using application-layer replication technologies (for example Replicas, Availability Groups, Dataguard) or application-layer intelligence (for example two-phase commit).

Micro-services

Another characteristic of cloud applications is the use of a modular ‘micro-services’ architecture in place of large, complex application systems. The benefits of micro-services revolve around simplicity — each module is simple to develop, deploy, test and consume — and scalability.

Although micro-services are simple to develop, they are often harder to orchestrate than a large, complex application. But although a traditional, monolithic application can initially be easier to orchestrate, it becomes more difficult and inflexible over time owing to the levels of integration and regression testing associated with even minor changes.

The use of micro-services also supports other characteristics of cloud applications such as loose coupling and horizontal scaling.

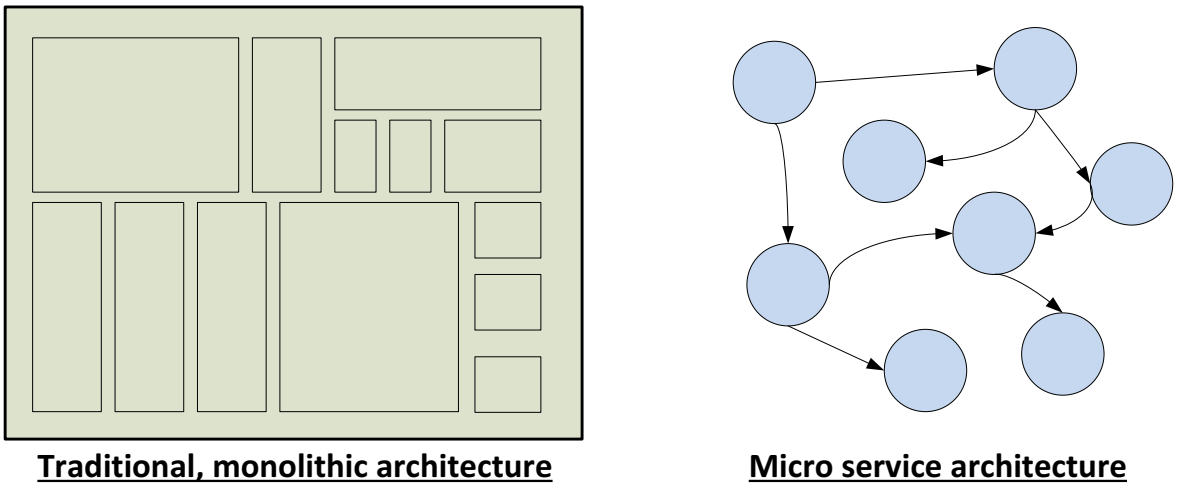

Figure 1 . Comparison of traditional and micro-services architectures

Loose coupling

Legacy and enterprise applications are typically tightly coupled. This means that if an individual layer of the application stack fails, so does the entire application. For example, an application with a tightly coupled database cannot continue to service requests if the database becomes unavailable; and an application using a thick client cannot operate if connectivity between the client and the server is lost.

Cloud applications should be designed to be loosely coupled so that, wherever possible, application components can continue to operate even if other application components fail or are unavailable. For example, a loosely coupled mobile app will allow the user to benefit from at least some functionality even if there is no network connectivity to the server.

Loose coupling can be achieved using a service-oriented architecture (SOA) in which each component interacts with other components via published interfaces (for example, REST API). Another approach is to implement message queuing. This lets components interact with each other by submitting messages via a message queue which is processed by the first available worker.

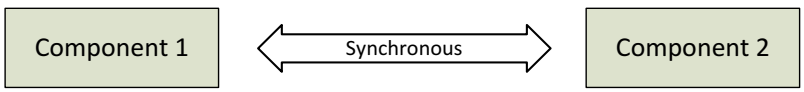

Figure 2. Traditional tight coupling

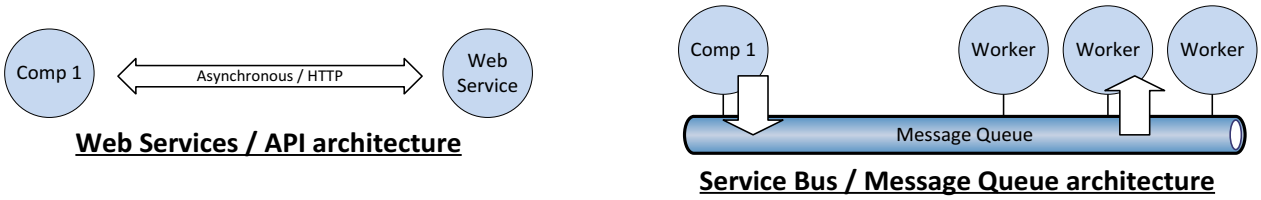

Figure 3. Approaches to loose coupling for cloud applications

Horizontal scale

Unlike legacy and enterprise applications, cloud applications are designed to scale horizontally — or scale out. The architecture calls for many small servers to work in parallel to service the requests — similar to grid computing. This contrasts with vertical scaling where workloads are restricted to a single server which is scaled up by adding resources such as CPU and memory. Horizontal scale provides the benefits of high performance and high availability at an affordable price point thanks to the elastic nature of cloud services. This is what enables customers to pay for large-scale solutions only during peak demand, scaling back when demand subsides to reduce on-going costs.

For cloud applications to dynamically benefit from horizontal scalability, they must be self-aware. They must be able to monitor the health and utilisation of the nodes so that they can orchestrate the underlying platform to add or remove nodes as appropriate — this is commonly referred to as auto-scaling. This feature is relatively easy to implement in the UKCloud Compute-as-a-Service environment, as an application can programmatically provision new virtual machines, reconfigure the virtual load balancer and shut down virtual machines via the UKCloud API.

Cloud agnosticism

Whereas legacy and enterprise applications are heavily dependent on the underlying infrastructure (for example, mainframe applications only work on certain mainframe hardware), cloud applications are much more portable across cloud platforms thanks to the inherent abstraction provided by the cloud. This enables applications to exist in virtual containers which are intentionally generic, so that workloads can run on a variety of hardware platforms and cloud service providers.

To prevent lock-in, cloud applications should use standards-based solutions wherever possible. For example, a number of UKCloud customers programmatically control the UKCloud environment via an open source abstraction technologies, examples include Fog — http://fog.io/ and Terraform https://www.terraform.io/.

Although this Blueprint focuses on Infrastructure-as-a-Service cloud platforms, organisations can achieve significant benefits by designing applications to leverage Platform-as-a-Service (PaaS), such as Cloud Foundry and OpenShift, the most popular Open Source PaaS offerings. Cloud Foundry and OpenShift can run on a variety of infrastructure solutions including vCloud Director–based IaaS, OpenStack platforms, VMware ESX platforms and Amazon Web Services. Indeed, PaaS makes it easier to run applications across multiple cloud service providers at the same time.

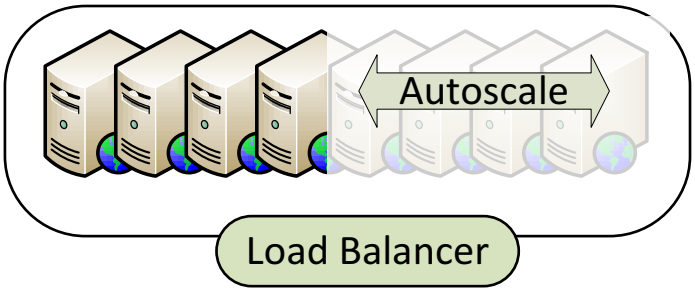

Figure 4. Scalability in the cloud

Stateless, shared-nothing architectures

Cloud applications are designed to expect occasional failures, and use a principle of redundancy to ensure multiple nodes can continue processing even if an individual node or an entire site becomes temporarily unavailable. This requires individual nodes to be designed to be stateless, which means that no persistent data is stored within the node. That way, if a node becomes unavailable, it has nothing specific to it which prevents other nodes from continuing processing.

Legacy and enterprise applications might achieve this by using shared storage solutions, in which a single data volume is connected to multiple nodes, so that all nodes have access to the data even if a single node fails. However, this approach brings a number of challenges:

- The shared storage represents a single point of failure — if the data volume should fail or become corrupted, all nodes would fail, too.
- This type of architecture typically requires expensive hardware components and additional software licensing.
- The architecture relies on tight coupling between the nodes and the storage which is not desirable in terms of scalability or resilience.
- Cloud applications will instead employ a shared-nothing principle in which each node is entirely independent and has its own data volumes to eliminate single points of failure. All stateful data is replicated across all nodes, either via application-layer replication technologies (for example, common in Hadoop, MySQL, Microsoft SQL database mirroring), or via application-layer data integrity components (for example, two-phase commit, where an application writes the data to two or more nodes before committing the transaction).
- It is well documented that shared-nothing architectures deliver superior performance and availability at scale. Many organisations are replacing traditional relational data stores (for example, Microsoft SQL Server, Oracle Database) with NoSQL (Not Only SQL), data stores (for example, MongoDB, Cassandra, Hadoop) and DBaaS (for example Trove) as they look to take advantage of the power of cloud platforms.

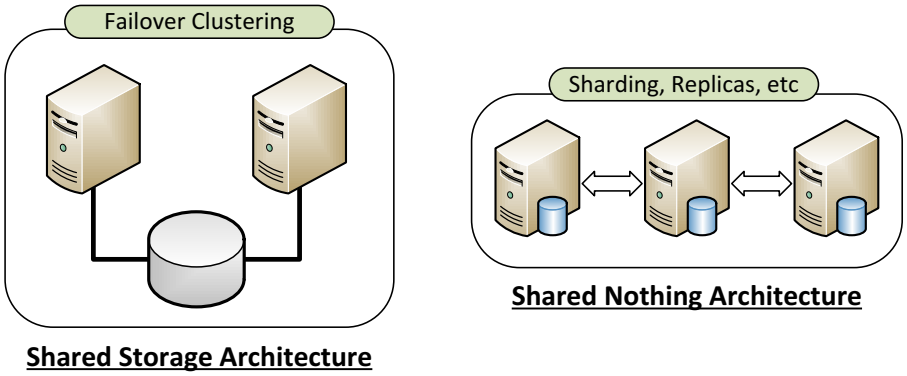

Figure 5. Comparison of shared storage and shared-nothing architecture

Scalable object storage

Traditional storage models such as SAN (block storage) and NAS (file storage) aren't well suited to the scalability, security and availability requirements of cloud applications. That's why a new storage model — object storage — has emerged which is optimised for cloud applications. It provides a much higher level of autonomy relative to traditional storage models because capacity, security and availability are defined and provided on a per-object, rather than a per-volume, basis.

Most importantly, object storage is inherently loosely coupled as it is controlled and accessed over a REST API. This makes it an ideal target for data such as web assets (including PDFs and videos) and large datasets such as archives and backups.

Our Cloud Storage is an object storage solution optimised to work with cloud applications. The solution is secure (suitable for all OFFICIAL data as it is NCSC Pan Government Accredited at Assured OFFICIAL and Elevated OFFICIAL), scalable, and available in both of UKCloud's UK data centres. The API is well documented, has a number of open source SDKs for most popular languages, and is compatible with the Amazon S3 API, which means it works with existing code, tools and applications that use Amazon S3.

Licensing and open source

When designing applications for the cloud it is important to consider the implications of using commercial software. Most commercial software runs well in the cloud, is supported by the software vendor, and benefits from cloud-friendly licensing schemes. However, some commercial software can't run in the cloud for one or more of the following reasons:

- Technology issues. Most software runs on standard Intel-compatible x86/x64 systems, as used in the cloud. However, some software requires specific technology such as mainframe, AIX or Solaris SPARC.

- Support issues. Most software vendors are comfortable supporting their software in virtualised and cloud environments, including complex enterprise applications such as SAP and Microsoft. However, some vendors provide limited support in the cloud which could be an important consideration for critical production services.

- Licensing issues. Some software is based on legacy licensing models which are not compatible with the cloud. For example, Oracle typically requires customers to

license all physical processors that the application could potentially be run on, which could clearly be an issue in cloud-scale deployments. Other software, for example Microsoft Windows desktop operating systems, may be licensed only to run on physical hardware dedicated to the customer. And some vendors, such as Apple, only license their software to run on their own branded hardware.

Awareness about these restrictions and limitations is driving many organisations to consider open source solutions in preference to commercial solutions. Open source software is typically more flexible in terms of licensing and compatibility. In addition, support is provided within the customer organisation or the wider open source community, rather than by a single commercial organisation. It's rare to find an open source solution that isn't supported in cloud and virtualised environments.

Cloud security: defence-in-depth, NCSC Cloud Security Principles

A common argument against cloud applications is the perception that legacy and enterprise applications are more secure. This perception stems from the fact that most legacy and enterprise applications are deployed in private environments inside the customer organisation. The concern is that if applications are hosted externally, the data held in those applications may become less secure.

It's a perception that has widely been challenged in recent times. The data centres and physical devices used within cloud servers are generally managed with a higher degree of security than organisations can implement internally.

For example, UKCloud's cloud platform operates from data centre facilities that can achieve both accreditation to host IL5 workloads and List X status. The security of cloud services is further validated through the NCSC G-Cloud Pan Government Accreditation Scheme. Under this scheme, UKCloud has achieved accreditation to host both Assured OFFICIAL and Elevated OFFICIAL workloads (and IL4 by aggregation).

Additionally, UKCloud has achieved PSN Accreditation to provide services at both 'PSN Assured service' (Assured OFFICIAL) and 'PSN Protected service' (Elevated OFFICIAL), demonstrating the depth of security implemented within our cloud platform.

NCSC has published Cloud Security Principles which provide guidance on to implement security within the cloud. UKCloud has implemented all 14 Cloud Security Principles and can advise its customers on how to operate securely within our assured cloud platforms.

EXAMPLE CLOUD APPLICATION ARCHITECTURE

This example involves a citizen-facing application that needs to deliver a highly available and scalable service. Typically, the application will be designed as a three-tier application:

- Presentation layer — responsible for handling web requests
- Application layer — responsible for processing the web requests and delivering application functionality
- Data layer — responsible for storing persistent data assets
- **Error! Reference source not found.**6 shows the example architecture featuring: an pplication running across two independent sites, an elastic presentation layer with auto-scaling web servers, a loosely coupled application layer, and a data layer with a distributed data architecture. Each layer is described in more detail below.

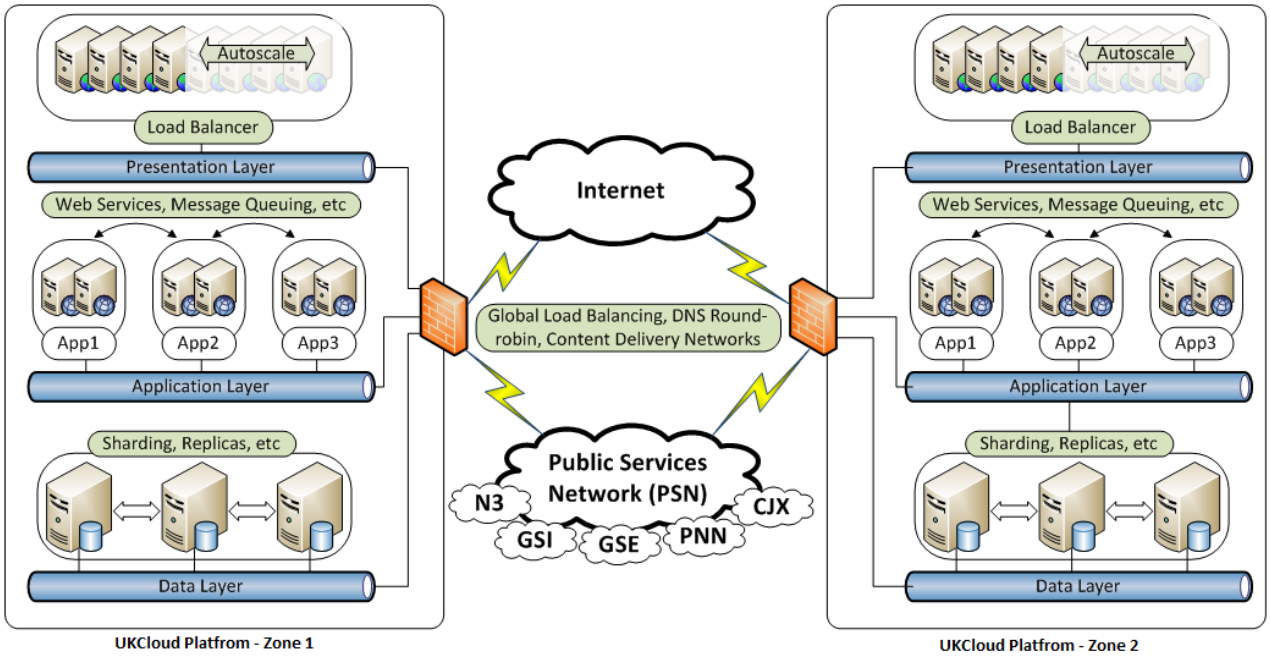

Figure 6. Schematic of example cloud application architecture

Presentation layer

The presentation layer is likely to have the largest demand for scalability and elasticity. When idle, the solution can probably run on just one or two web servers. But at peak times the application may need to support millions of users, so the presentation layer will need to scale out to dozens or hundreds of servers.

UKCloud recommends that its customers use relatively small virtual machines for this purpose as it makes more sense to run many, smaller servers rather than fewer, larger ones. VMs without automatic back-ups are easily re-provisioned, rather than recovered from backup.

A load balancer is required to distribute requests across the servers. The one included in the vShield Edge virtual firewall could be used, but there may be more benefit in using a virtual appliance from a specialist such as F5 Networks. That's because these virtual appliances typically provide additional functionality such as more intelligent load-balancing algorithms, health and utilisation monitoring (to identify when to scale out and scale back), and workflows (to easily add and remove servers from the load-balancing farm).

Application layer

This layer provides the functionality of the application. It’s likely there will be various types of application server at this layer. For example, some might provide searching and indexing function, some might provide reporting and analytics function, and others the content management function.

For bespoke application components, it's worth taking a similar approach to the presentation layer, using smaller servers. For pre-built application components, like Alfresco and Lucene, the applications' requirements should be followed. Some components may need to scale up — in which case larger servers can be used; while others might need the Automated VM Backup and data replication as required.

If possible the application component should use loose coupling when interfacing with other components, including the data layer. Implementing message queuing, service bus or web services enables this to be achieved in a scalable, highly available and secure manner.

Data layer

This is the layer at which the data which drives the application is persistently stored. Traditionally, this is the most secure layer of the application, to maintain the confidentiality, integrity and availability of sensitive data.

The most common traditional data store is a relational database management system (DBMS) such as Microsoft SQL Server, Oracle or DB2. All of them need very powerful, reliable server hardware, and often require expensive licenses to achieve high availability through clustering technology.

UKCloud recommends that its customers consider alternative data stores which are much better suited to modern platforms such as cloud. SQL data stores such as MySQL and PostgreSQL work effectively in the cloud — especially using techniques such as data sharding. However, the real benefit comes from scale-out, shared-nothing data architectures such as NoSQL solutions like MongoDB, Riak, Cassandra, or a distributed data fabric such as Hadoop.

With NoSQL solutions, the servers required to support the application will be similar to the application and presentation layers.

ABOUT UKCLOUD

UKCloud is dedicated to the UK Public Sector. We provide assured, agile and value-based true public cloud that enable our customers to deliver enhanced performance through technology.

- **We're focused on cloud.** Delivering a true cloud platform that is scalable, flexible, assured and cost-effective.
- **We're open. You are never locked in.** Using industry standards and open source software we enable flexibility and choice across multiple cloud solutions.
- **Dedicated to the UK Public Sector.** Our business is designed specifically to serve and understand the needs of public sector organisations.
- **We develop communities.** We bring together communities of users that are able to share datasets, reuse code, test ideas and solve problems.
- **Customer engagement.** We will only be successful if our customers are successful. We embody this in the promise: Easy to adopt. Easy to use. Easy to leave.

Additional information about UKCloud can be found at www.ukcloud.com or by following us on Twitter at @ukcloudltd.

UKCloud. The power behind public sector technology.

MORE INFORMATION

For further information about UKCloud and how we can help you, please email us at info@ukcloud.com.

UKCloud Ltd

A8 Cody Technology Park
Ively Road, Farnborough
Hampshire, GU14 0LX

T 01252 303300
E info@ukcloud.com

www.ukcloud.com


@ukcloudltd


ukcloudltd


ukcloud-ltd

UKC-GEN-132 • 08/2017 • version 2.0