Deploying highly available and secure cloud solutions

December 2012

Deploying highly available and secure cloud solutions

Authors and contributors

DAVID BILLS – Microsoft Trustworthy Computing

CHRIS HALLUM – Microsoft Windows

YALE LI – Microsoft IT

MARC LAURICELLA – MicrosoftTrustworthy Computing

ALAN MEEUS – Windows Phone

DARYL PECELJ – Microsoft IT

TIM RAINS – Microsoft Trustworthy Computing

FRANK SIMORJAY – Microsoft Trustworthy Computing

SIAN SUTHERS – Microsoft Trustworthy Computing

TONY URECHE – Microsoft Windows

Table of contents

Executive summary

Many organizations today are focused on improving the flexibility and performance of cloud applications. Although flexibility and performance are important, cloud applications must also be available to users whenever they want to connect. This paper focuses on key methodologies that technical decision makers can use to ensure that your cloud services, whether public or private, remain available to your users.

At a high level, each cloud session consists of a customer using a computing device to connect to an organization's cloud-based service that is hosted by an internal or external entity. When planning for a highly available cloud service, it's important to consider the expectations and responsibilities of each of these parties. Your plan needs to acknowledge the real-world limitations of technology, and that failures can occur. You must then identify how good design can isolate and repair failures with minimal impact on the service's availability to users.

This paper showcases examples for deploying robust cloud solutions to maintain highly available and secure client connections. In addition, it uses real-world examples to discuss scalability issues. The goal of this paper is to demonstrate techniques that mitigate the impact of failures, provide highly available services, and create an optimal overall user experience.

Introduction

Customers have high expectations for the reliability of computing infrastructure, and the same expectations apply to cloud services. Uptime, for example, is a commonly used reliability metric. Today, users expect service uptimes from 99.9% (often referred to as three nines) to 99.999% (five nines), which translates to nine hours of downtime per year (at 99.9%) to five minutes of downtime per year (at 99.999%). Service providers frequently distinguish between planned and unplanned outages, but, as IT managers well know, even a planned change can result in unexpected problems. A single unexpected problem can put even a 99.9% service commitment at risk.

Reliability is ultimately about customer satisfaction, which means that managing reliability is a more nuanced challenge than simply measuring uptime. For example, you can imagine a service that never goes down but that is really slow or that is difficult to use. Although maintaining high levels of customer satisfaction is a multifaceted challenge, reliability is the foundation upon which other aspects of customer satisfaction are built. Cloud-based services must be designed from the beginning with reliability in mind. The following principles of cloud service reliability are discussed in this paper:

- Use a service-oriented architecture
- Implement separation of function
- Design for failure
- Automate testing and measurement
- Understand service level agreements

Measuring reliability and user expectations

In addition to uptime, which was discussed earlier, other reliability metrics exist that should be considered. A common measurement of computer hardware reliability is *mean time to failure* (MTTF). If a component fails, the service it provides is unavailable for use until the component is repaired. However, MTTF only tells half of the story. To track the time between failure and repair, the industry created the *mean time to repair* (MTTR) measurement. To calculate an important metric of service reliability, we can use the equation of MTTF/MTTR. This equation shows that reducing the repair time by half will result in a doubling of the measured availability. For example, consider the situation of an online service that has historically demonstrated an MTTF of one year and an MTTR of one hour. In terms of measured availability, halving the MTTR to a half hour is equivalent to doubling the MTTF to two years.

By focusing on MTTR, you can mitigate the potential impact of failure incidents and seek to improve reliability by creating a set of standby servers with a sufficiently redundant design to hasten recovery from such incidents. You should always document these types of mitigations in a service level agreement (SLA) from the cloud provider. By documenting them you are implicitly acknowledging that some amount of failure is expected to occur, and that the best way to minimize the impact from failure is to increase the MTTF and reduce the MTTR.

The following sections detail some of the key architectural requirements of designing highly available cloud-based services.

Service-oriented architecture

Effective cloud technology adoption requires appropriate design patterns. In a service-oriented architecture, each component should have a well-designed

interface so that its implementation is independent of every other component and able to be used by new components as they are deployed. Designing architecture in this way helps reduce overall system downtime, because components that call into a failing component can properly handle such an event.

Separation of function

Separation of function, also known as separation of concerns, is a design pattern that states that each component will implement only one or a small set of closely related functions with no overlap and loose coupling to other components. The three-tier architecture shown in Figure 1 later in this paper is a classic example of separation of function. This approach allows functionality to be spread across different geographies and networks so that each function has the best chance to survive failures of specific servers. The figure depicts redundant front-end web servers, message queues, and storage, each of which could be separated geographically.

Automatic failover

If the component interfaces are registered with a uniform resource identifier (URI), failover to alternate service providers can be as simple as a DNS lookup. Using URIs instead of locations for services increases the likelihood that a functioning service can be located.

Fault tolerance

Also known as graceful degradation, fault tolerance depends on aggregating the building blocks of a service without creating unnecessary dependencies. If the user web interface is as simple as possible and decoupled from the business logic or back-end, the communications channel can survive failures of other components and maintain the organization's link to the user. In addition, the web interface can be used to inform the user of the current status of each piece of the organization's cloud-based service. This approach not only helps the user understand when to expect full restoration of services, but also improves user satisfaction.

Disaster planning

You should expect that services will fail from time to time. Hardware failure, software imperfections, and man-made or natural disasters can cause service failure. You should complete planning for routine problems before deployment to help troubleshooters know what to look for and how to respond. But even huge environmental disruptions, sometimes known as *black swan* events, will occur periodically; therefore, you need to consider such events during the planning process. The black swan theory posits that these unlikely events collectively play vastly larger roles than regular outages.

Test and measure

Two types of test and measurement of a running service are appropriate. The automated polling of the service by a test server can result in early detection and reporting of failure and thereby reduce the MTTR.

User research should be conducted either immediately before or after a deployment to understand how users react and identify unmet expectations. An easy way to obtain user feedback is to simply ask them for it, not every time you see them but occasionally. You should be able to get useful data, even with a very low response rate, and you will also obtain key performance indicators (KPIs) from users for monthly status reports.

Cloud provider

Two concepts from the 1970s have been realized by new technologies in today's cloud offerings.

- Virtualization of computer hardware is a reality, with virtual computer images and virtual hard drives that can be remotely managed.
- Fast scaling and agility are realities, with management tools that can control the power of physical and virtual hardware.

It's important that IT professionals understand these concepts and their powerful capabilities. As stated in the "Executive summary" section, the concepts in this paper apply to both public and private clouds, each of which has a place in the toolbox of forward-looking IT departments. The focus from the start of any project should be on cloud design and management services that provide minimally disruptive service delivery to users.

Cloud provider expectation and responsibility

A natural shared responsibility exists between any organization and its chosen cloud provider. For custom applications, the cloud provider designs its service for reliability based on the use of certain features, such as failover and monitoring, by the developer. The developer must understand and use these features for reliability to be an achievable goal.

An organization that implements a solution on top of cloud-based infrastructure must ensure that the service is available as much as possible. Users expect these types of services to be as reliable as a telephone. Outages may occur, but they are rare, localized events. The organization's ability to provide such assurance requires transparent communication with the provider about what to expect from the service and what must be supplied by the service consumer. Without such transparency, finger-pointing about responsibility can occur instead of automated recovery when service failures lead to outages for users.

The same consideration applies to private clouds. An IT organization can partition infrastructure responsibility to in-house experts who then create a private cloud. The cloud service is expected to provide a reliable platform on which the rest of the IT team can create innovative solutions to address the business needs of the organization. But just as for an outsourced cloud service, fully transparent operations and documentation of expectations will help avoid failures that result from vague or poorly defined areas of responsibility.

Cloud availability

Typically, public cloud services provide high availability by using a geographically distributed and professionally managed collection of server farms and network devices. Even very large enterprises that have private clouds for specific high-value content can profitably use public cloud services to host their application solutions. And large public clouds have effectively infinite capacity, because they can respond with more servers when demand is greater than anticipated. Offloading the excess server capacity has multiple benefits, the most prominent of which is that, in most public cloud models, excess capacity is not billed until it is used.

Many cloud providers offer built-in capabilities for increased availability and responsiveness, including:

- Round-robin DNS
- Content distribution networks
- Automated failover
- Geographic availability zones

Design for availability

As stated earlier, the most effective way to increase availability is to shorten the MTTR. If geographic and network diversity is available from the cloud, ensure that load balancing automatically routes users away from failed components to working components. Even a relatively simple capability such as network load balancing can be affected by unexpected interaction between the organization

and the cloud provider or DNS. Such interactions have the potential to introduce instabilities in the service offering that have not been anticipated.

For example, distributed denial of service (DDoS) attacks are external attacks against availability that all cloud services need to mitigate. However, unless mitigation is carefully implemented, organizations with little security experience can unintentionally cause an application to become unavailable, which can result in as much downtime damage as a DDoS attack. DDoS mitigation is an example of a capability that is most effectively provided at scale—that is, by the cloud provider or the ISP.

Most major cloud vendors are certified for reliability and security, which they report in documents such as those contained in the Cloud Security Alliance (CSA) Security, Trust, and Assurance Registry (STAR).[1] Although STAR itself is relatively new, it's important for IT managers to consider that most in-house systems have not received such third-party vetting. Obtaining such assurance at a shared cost is another potential benefit of public cloud services.

When evaluating the benefits and challenges of creating a highly available cloud-based solution, it is important to ensure that your design includes a threat analysis of well-known problems such as those defined earlier, as well as any business-disrupting failures that are unique to the solution you want to deploy. Typically, only security attacks are considered in a threat analysis, but a well-designed cloud solution will consider other types of loss of availability and plans for mitigations as well.

The following figure shows a generic three-tier design with redundancy capability. Each request has more than one path to a component that can respond to it. Starting from the left is a user device, such as a laptop computer, from which the request is routed (again, consider the use of round-robin DNS and network load balancing features, if available). In tandem, an automated availability test service is exercising as much of the system as possible to assure that any failure is quickly reported so that necessary repairs can begin quickly.

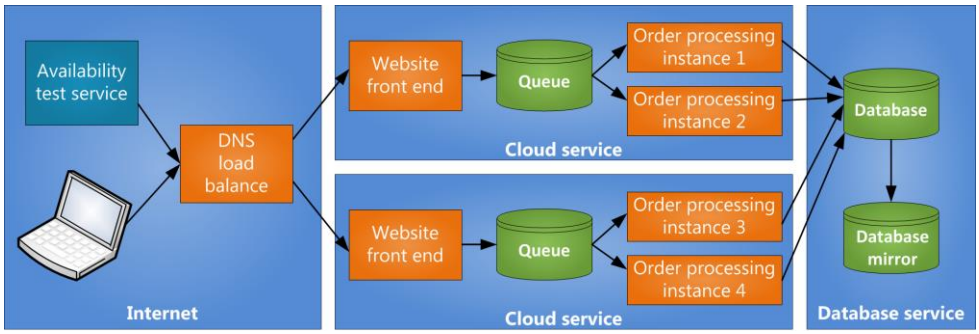
The cloud service in the figure shows separation of function. This separation helps ensure that the network path to the user has no common components that result

[1] Security, Trust and Assurance Registry (STAR), at https://cloudsecurityalliance.org/star/

in failure of both paths. Within each cloud service site, additional component redundancies are possible. In this scenario, the queue may be provided by the cloud service provider. If a common database needs to be shared between both sites, it is up to the organization's application architecture to route traffic to the data in a way that will fail over to some other site running a mirrored copy of the database; however, many public cloud offerings have built-in data redundancy capabilities that should also be explored.

In this example solution, load balancing is split between the Internet at the front end, the cloud service in the middle tier, and the organization's application at the back-end. Tests of disabling each of these components should be undertaken with live loads to assure that fail-over options work as planned.

Figure 1. Designing for availability



If the website front-end component is sufficiently simple and straightforward, users should always be able to see the enterprise brand image and status information, which will help them have confidence in the reliability of the enterprise itself. Simplicity in both components and connections is the key to the reliability and availability of the system as a whole. Complex and tightly interconnected systems are difficult to maintain and debug when something fails.

Organizational customer of the cloud

An organization that acquires cloud services from either private or public cloud providers needs to fully understand the responsibilities of the cloud provider as well as the limitations of those responsibilities. Similarly, the cloud provider must understand the availability and security requirements of the solution that it provides to users. A complete cloud solution requires a thoroughly reliable implementation and an ability by the cloud service provider to create a service that integrates its own capabilities with the organization's requirements. The good news is that this integration is where the most innovation and value-add for the organization is generated.

The organization's responsibility

When the responsibilities of the cloud provider are specified, well understood, and documented in a service level agreement, any unmitigated threats become the responsibility of the customer—the organization. A best practice for identifying potential unmitigated threats is to conduct a brainstorming session to identify all possible threats and then filter out those that are known to be the responsibility of the cloud provider. Threats that remain are the organization's responsibility to mitigate. The following list of risks and responsibilities can be used as a guide for types of threats to consider:

- **Apply access control locally and in the cloud**. Although data loss incidents can occur for a variety of reasons, they most commonly occur when an attacker either spoofs the identity of a valid user or elevates their own privileges to acquire access that has not been authorized. Most organizations compile a directory of employees and partners that can be federated, or have their accounts mirrored, into a cloud environment; however, other users may need to be authenticated using different methods. For future flexibility, adopt

cloud services that support federation and accept identities from on-premises directories as well as from external identity providers. The trend is for providers to include access control services as a part of their service offering. A best practice is to avoid duplicating your account database, because doing so increases the attack surface of the information it contains, such as password data.

- **Protect data in transit**. Data loss can occur if the data is not protected in storage or in transit. Protecting data in transit can be accomplished by using Transport Layer Security (TLS) to provide encryption between endpoints. Protecting data in storage is more of a challenge. Encryption can be provided in the cloud, but if the data is to be decrypted by apps that also run in the cloud, the encryption key needs special protections. It's important to note that providing the cloud with access to the encryption keys as well as to the encrypted data is equivalent to storing the data unencrypted.
- **Protect trusted roles**. Authorization to perform administrative functions or to access high-value data will likely be based on users' roles within their organizations. Because roles will vary for each user while their identity remains constant, some mapping must exist between each user ID and a list of the ID's authorized roles. If the cloud provider is trusted to control access, this list must be made available to the cloud and managed accordingly. A best practice is to use claims-based authorization technologies such as Security Assertion Markup Language (SAML).
- **Protect data on mobile devices**. Protection of user credentials and other sensitive data on mobile devices is only feasible if security policy can be enforced. A best practice is to configure Microsoft Exchange ActiveSync mailbox policies. Although not all devices implement all of the ActiveSync policies, the market is responding to this need and organizations should seek to deploy and enforce a endpoint security solutions on all mobile devices.
- **Develop all code in accordance with SDL**. Application code is likely to come from a combination of the cloud provider (for example, in the form of sample code), the cloud tenant organization, and third parties. A threat modeling process such as the one used as part of the Security Development Lifecycle (SDL), the software development security assurance process created by Microsoft, needs to consider this factor. One area in particular that needs to

be analyzed is the potential for conflict if more than one component controls related functionality, such as authorization.

- **Optimize for low MTTR**. The threat modeling process also needs to consider and specify different types of expected failures to help ensure low MTTR. For each potential failure, specify the tools and technologies that are available for recovering functionality quickly.

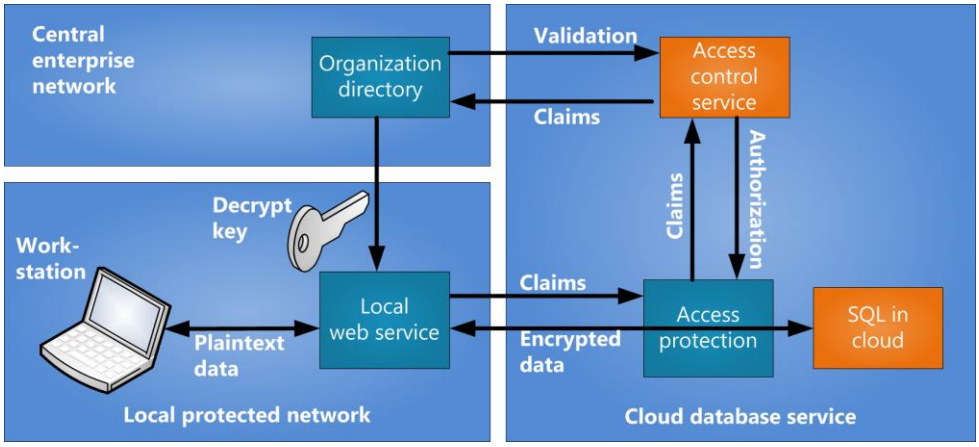
Availability of sensitive information stored in the cloud

The cloud offers some interesting options for securing data within a highly available architecture. Consider the following example cloud solution, in which the functionality for both security and availability is split between the organization and the cloud. Sensitive organizational information is stored in the cloud, but the decryption keys are maintained within the organization so that no attack on the cloud can reveal the sensitive information. One option is to encrypt all data that is stored in the cloud to prevent data leakage. However, it is also possible to differentiate between types of data so that high-value data is protected by encryption and low-value data is protected only by access control. The design principles of separation of function and geographical distribution that were suggested earlier are used here to increase resiliency.

The following figure illustrates how the data protection scenario works. Data is entered and retrieved at a workstation that is attached to an organization's network. The network uses a firewall to protect it from intrusions. The workstation connects to a service in the network that uses a key to encrypt and decrypt data. The key is obtained from the organization's central directory, so all distinct local protected networks can access the data that is stored in the cloud.

The user is authenticated by the organization's directory. Federation and SAML are used to provide centralized control of authentication and authorization, which takes advantage of the available existing account repository (such as Active Directory) in a distributed environment.

Figure 2. Data encrypted in the cloud



This example of encrypted data in the cloud can be used as a pattern for a variety of implementations. The protection of the data decryption key removes the threat of data leaks in the cloud and puts control of the plaintext data under the organization's full control. The distributed architecture helps to ensure the availability of the service.

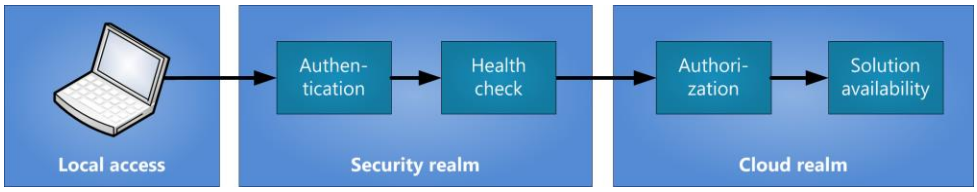
The user and the device used to access the cloud

Users will measure the availability of a cloud service solely in terms of their ability to complete their current task, which means that the cloud service as well as the device that they use to access the cloud must be functional.

User expectation and feedback

Users measure service availability based on their success at achieving their objectives. The following figure shows the typical steps in the process of a user obtaining access to a cloud resource. First, the user's device must connect to the local network and be authenticated. Next, the device's security disposition, or health, is checked and some sort of role-based authorization process is used to establish appropriate access for the user. Finally, the cloud resource itself must be available. The failure of any of these components will block the user's ability to complete their task. Sometimes the blockage is desired for security purposes, but the user will always perceive it to be an impediment.

Figure 3. Availability blockers



When any one of the links shown in the figure fails, it is important to let the user know the nature of the problem and also what needs to be done to restore availability of the solution. Whenever user action is required, instructions need to be clear and concise.

Design for test

An automated test program is helpful for detecting solution failures. A best practice is to design the solution for online testing. All customer-facing services and webpages need to enable automated query programs that use near real-time reporting with automated escalation when significant failures occur. Online testing can provide valuable performance indicators of the availability of the solution services.

Some method of communicating users' perception of availability should be implemented as well. For example:

- If an application provides users with access to the cloud, use it to generate statistics, such as time from login to acquisition of cloud data.
- Periodically ask users when their session ends if they would take a short survey.
- Send user experience researchers into the field to get user feedback.

User device availability

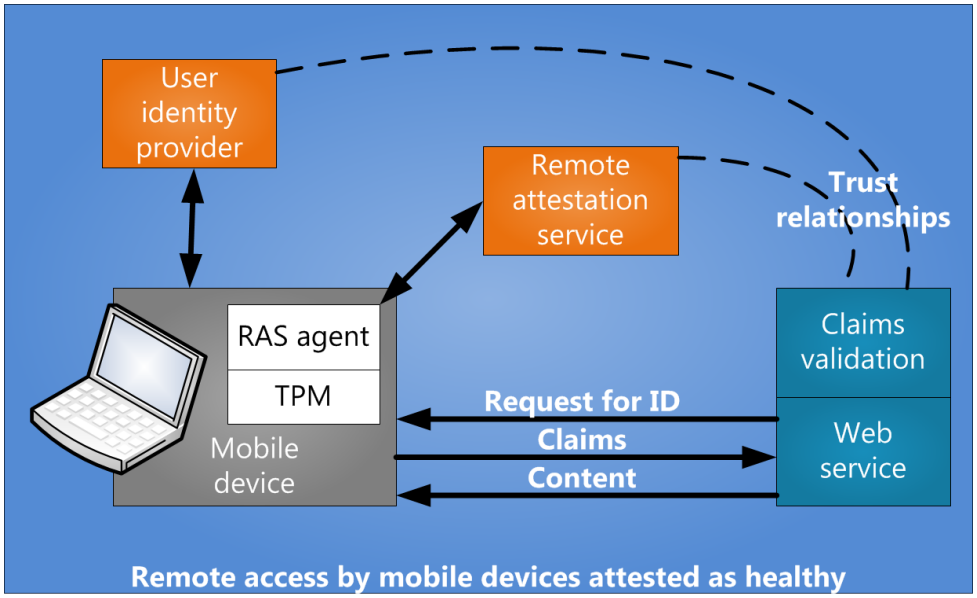
Devices that are used to access cloud solutions must be trusted not to leak high-value information. Because mobile devices are increasingly being used to access such information, some way to evaluate device security, or health, is required. A good cloud design is able to evaluate device health and verify user identity. This section describes how to provision device health assessment in such a way that the cloud solution can be available to users from anywhere.

This example solution addresses the need to establish secure access to an organization's network from a user-owned device. This type of scenario is often referred to as bring your own device to work, or BYOD. For many years, IT departments were able to protect enterprise assets by quarantining all resources, including the client computers that accessed those resources, inside a protected perimeter. In BYOD scenarios users have commercially available devices that can access all of their personal data from anywhere, and they want to use those same devices to access the organization's resources as well. This phenomenon is known as the consumerization of IT.

In such a scenario, the cloud service needs to establish the user's identity, learn their preferences about how their personal information can be used, and obtain the user's permission if the service (or the organization) wants to store the user's personal information for future use or share it with others. In addition, the cloud service may have content that should only be released to user devices that are determined to be secure. Many users want to know that their privacy, identity, and assets are protected from malware, although most users are unwilling to be inconvenienced by security mechanisms.

The following figure shows a solution built to assess mobile device health from the cloud. The mobile device authenticates the user through a connection to an identity provider in the cloud. If the web service has highly confidential information, or is trying to obtain a provable indication of the user's intent, it may elect to verify the security of the mobile device before proceeding. The user's device will then receive a health attestation that can be sent with the user ID.

Figure 4. Secure mobile clients



Windows 8 devices can be protected from low-level rootkits and bootkits by using low-level hardware technologies such as secure boot and trusted boot.

Secure boot is a firmware validation process that helps prevent rootkit attacks; it is part of the Unified Extensible Firmware Interface (UEFI) specification. The intent of UEFI is to define a standard way for the operating system to communicate with modern hardware, which can perform faster, more efficient input/output (I/O) functions than older, software interrupt-driven BIOS systems.

Trusted boot creates a condition in which malware—even if it is able to tamper with the boot process, which is unlikely—can be detected, which prevents a health attestation from being granted. Secure boot also protects the antimalware software itself.

A Remote Attestation Service (RAS) agent can communicate measured boot data that is protected by a Trusted Platform Module (TPM). After the device successfully boots, boot process measurement (for example, measured boot in Windows 8) data is sent to a RAS agent that compares the measurements and conveys the health state of the device—a positive, negative or unknown state—by sending a health claim back to the device.

If the device is healthy, it passes that information to the web service so the organization's access control policy can be invoked to grant access.

Depending on the requirements of the content provider, device health data can be combined with user identity information in the form of Security Assertion Markup Language (SAML) or open standard for authorization (OAuth) claims. The identity provider, for example Active Directory, may belong to the user's employer, to the content provider, or to a social network such as Facebook. The data is evaluated by fraud detection services that are already in use at most commercial websites. Access to content is then authorized to the appropriate level of trust for what the health assertions, or claims, merit. These claims protocols are structured to allow additional requests from the content provider to the user's device as needed by the user's transaction requests of the provider. For example, if high-value data or funds transfers are requested, additional security state may need to be established by querying the user's device before the transaction can be completed.

Conclusions

The preceding examples illustrate solutions that emphasize a secure, service-oriented architecture with separation of functions. The demonstrated architectural patterns divide the solution into components with loose coupling. This approach allows each component to fail over gracefully, even if other components fail catastrophically. The service as a whole may continue with some or all functionality no matter which individual component fails. The design can use hybrid solutions that include some on-premises functionality while providing other functionality, such as solution scaling, through an off-premises public cloud.

As a best practice, availability needs to be monitored by a service that operates in the same realm as the user. In addition, services should be designed and located in a way that makes them accessible and operable from geographically diverse locations to provide availability when calamities and natural disasters occur.

Cloud service developers and cloud service customers alike need to communicate and cooperate to anticipate, design, and test for failures at every point. Such communication and cooperation will have a direct impact on the success of the solution and the satisfaction of its users.

Additional reading

For more information about the scenarios and solutions detailed in this paper, see the following resources. These documents provide additional information to help you make the right design decisions for the availability of your cloud-based solutions.

- The Windows Azure Application Model https://www.windowsazure.com/en-us/develop/nodejs/fundamentals/application-model/
- Microsoft System Center http://microsoft.com/systemcenter (this website is now focused on Release Candidate 2012)
- Cloud Computing: Achieving Control in the Hybrid Cloud http://technet.microsoft.com/en-us/magazine/hh389788.aspx
- Cloud Security Alliance - Security, Trust & Assurance Registry (STAR) https://cloudsecurityalliance.org/star/
- Security Guidelines for SQL Azure http://social.technet.microsoft.com/wiki/contents/articles/1069.security-guidelines-for-sql-azure.aspx
- Service-Oriented Architecture – Design Patterns www.soapatterns.org/masterlist_c.php
- Cloud Insecurity: Not Enough Tools, Experience or Transparency www.technewsworld.com/story/74890.html
- How the Cloud Looks from the Top: Achieving Competitive Advantage In the Age of Cloud Computing (PDF) http://download.microsoft.com/download/1/4/4/1442E796-00D2-4740-AC2D-782D47EA3808/16700%20HBR%20Microsoft%20Report%20LONG%20webview.pdf

One Microsoft Way
Redmond, WA 98052-6399
microsoft.com/twcnext