Creating Resilient Architectures on Microsoft Azure

Table of Contents

Introduction

RESILIENCY: *noun*

1. the power or ability to return to the original form, position, etc., after being bent, compressed or stretched; self-healing.
2. the ability to recover readily from failure, faults, user error, overuse or the like.

Today, just about everyone needs instant access to applications, compute and data services. The number of people connected, and the devices they use to connect to these services, are ever growing. And in this world of always-on services, the systems that support them must be designed to be both highly available and resilient.

When we ask customers if their applications and infrastructure are resilient and built to handle both temporary and large-scale failures, most will answer "yes." But before you answer that question for your business, ask yourself these three questions:

- What are the defined service level agreements (SLAs) for your applications? Do you have SLAs for all the workloads that contribute to your application?
- Have you mapped the failure points and can you quantify the business impact a particular failure might have?
- Do you regularly measure the health of your infrastructure and applications?

Chances are, you probably can't answer "yes" to all of these. That's because a resilient application and infrastructure need a great deal of upfront planning and architecting, and require ongoing analysis and management throughout its lifecycle. Often, these needs aren't addressed until there is an issue because of time or resource constraints.

But in the cloud, it is more critical than ever that applications and infrastructure are designed and operated with an eye toward resiliency at every step. Your applications and services will be deployed in cloud environments that are populated with hardware and infrastructure in pre-defined configurations. Historically, you may have purchased higher-end hardware to scale up. But in cloud environments, you must scale out instead. You keep costs for these cloud environments low by using commodity hardware. Commodity hardware will fail, and the cloud requires the architecture to truly embrace failure. In the past, you may have focused on preventing failures and optimizing "meantime to fail." In the Azure environment, the focus shifts to one of "meantime to restore."

In the cloud, we must change our thinking. Things are going to fail, and you need to build that into your plan and design for failure. Your applications now have dependencies on things outside your control like hardware in the data center or outside application services or an internet connection. Now your focus is on identifying what can potentially go wrong and how your application will handle those failures.

Planning for failures and disasters in the cloud requires you to understand what can cause an outage, recognize the failures quickly and have a plan to restore service when something does go wrong. You then implement a strategy that matches your tolerance for the application's downtime. Additionally, you have to consider the extent of data loss the application can tolerate without causing adverse business consequences once it is restored.

In this paper, we'll walk through an overview of planning, implementing and operating resilient architecture in an Azure environment, and share some real-world examples.

What is a resilient architecture?

Usually when people talk about a resilient architecture there are two perspectives. The first is a hardware perspective, and the second is an application (software or service) perspective. In today's cloud environment you need to look at a combination of both to be truly resilient.

A resilient architecture should:

- **Maximize service availability for consumers** – ensure customers can access and use the service
- **Minimize the impact of a failure on consumers** – degrade gracefully, isolate faults and fallback to alternate delivery paths
- **Maximize performance and capacity** – provide services that are available and can handle the desired/required demand

One company that embraces resiliency for their online service is Netflix. By embracing failure and frequently testing its ability to recover from outages of all kinds, Netflix is better prepared for the inevitable failures that will occur and to minimize the business impact. For example, during an outage at its cloud provider, Netflix utilized an older video queue for customers when the primary data store was not available.

Another example of resilience would be an ecommerce site that can continue to collect orders if its payment gateway is suddenly unavailable. This resiliency may manifest itself as the ability to process orders when the payment gateway is once again available or after failing over to a secondary payment gateway.

The most common characteristics of a resilient architecture include:

- Fault tolerance
- Availability
- Scalability
- Self-healing
- Automation
- Redundancies
- Reporting

FAULT TOLERANCE

Resilient applications assume that every dependent cloud capability can and will go down at some point in time. A fault tolerant application detects and maneuvers around failure points to continue and return the correct results within a specific timeframe. A fault tolerant system will employ a retry policy for intermittent or transient error conditions. For more serious faults, the application can detect problems and fail over to alternative hardware or contingency plans until the failure is corrected. A resilient application will handle the failure of one or more parts and continue operating properly. Fault-tolerant applications may use one or more design strategies, such as redundancy, replication or degraded functionality.

AVAILABILITY

A resilient application understands the availability of its underlying infrastructure and dependent services, and removes single points of failure through redundancy and design. In Azure, we focus on the concept of effective availability of the overall application or service. Effective availability considers the service level agreements (SLA) of each dependent service and their cumulative effect on the total system availability. The more moving parts within the system, the more care you must take to ensure the application can meet the availability requirements of its end users.

SCALABILITY

Scalability directly affects resiliency. An application that fails under increased load is no longer available. Scalable applications are able to meet increased demand with consistent results in acceptable time windows.

When an application is scalable, it scales horizontally or vertically to manage increases in load while maintaining consistent performance. Horizontal scaling adds more servers or VMs (virtual machines) of the same size (processor, memory, and bandwidth), while vertical scaling increases the size of the existing machines.

For Azure, there are vertical scaling options for selecting various machine sizes for compute hosting. However, changing the machine size requires a time consuming re-deployment. Consequently, the most flexible solutions are designed for horizontal scaling. This is especially true for compute hosting because you can easily increase the number of running instances of any web or worker role and it also works well with services such as Azure Storage, which do not provide tiered options for vertical scaling.

SELF-HEALING

Self-healing describes any application or service that has the ability to recognize that it is not operating correctly and, without human intervention, make the necessary adjustments to restore itself to normal operations. For example, if you have a server that is leaking memory, disk space or other resources, a self-healing service could automatically reboot or re-image the server.

AUTOMATION

Automation provides a method for eliminating downtime due to human error. The more automation you can build into your resilient architecture, the less likely your application will be subject to downtime.

REDUNDANCIES

With Azure, deploying and managing redundant systems is handled by Microsoft from a hardware perspective. You'll need to focus your planning and resources more on the software side. Designing a redundant application is critical.

REPORTING

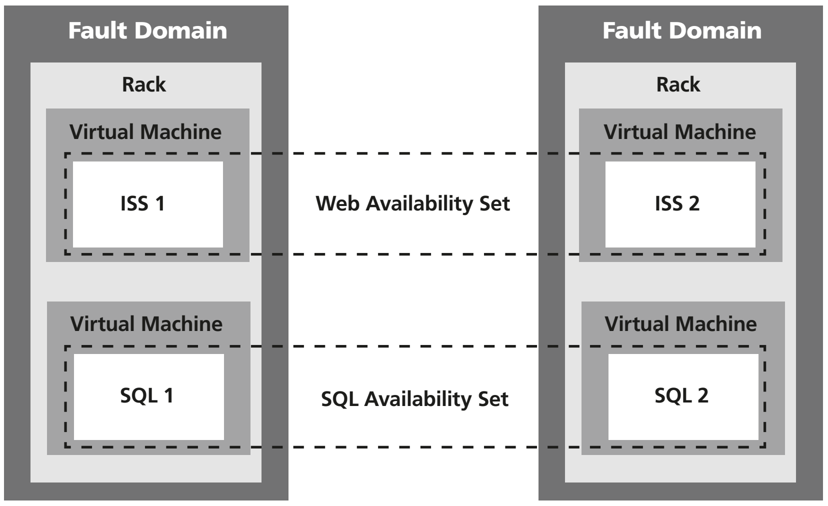
Being able to measure the health of your systems on a regular and ongoing basis is a step that is often overlooked. Understanding the performance and health of your applications not only helps provide a positive user experience, but also helps ensure that your application is available and meeting the pre-defined SLAs.

Azure strengths in a resilient architecture

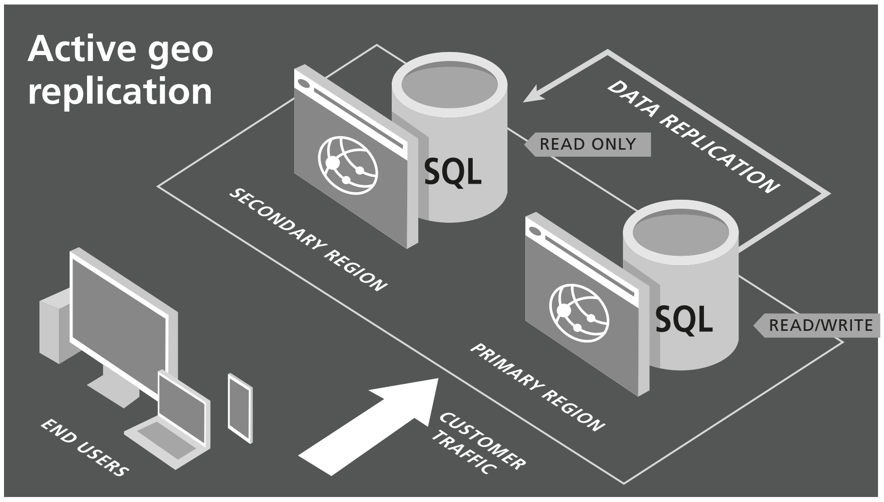
An architecture can only be as resilient as its base components. Microsoft built Azure from the ground up to provide a resilient platform for your applications and services using a combination of baseline platform technologies and configurable features. The following are key features of Azure that give you the ability to build a truly resilient architecture.

Azure Fabric Controller (FC): The Azure Fabric Controller is responsible for provisioning, delivery and monitoring of Azure compute instances. The Fabric Controller checks the status of the hardware and software of the host and guest machine instances. When it detects a failure, it enforces SLAs by automatically relocating the VM instances.

Fault Domains: When multiple role instances are deployed, Azure deploys these instances to different Fault Domains. A Fault Domain boundary is basically a different hardware rack in the same data center as depicted in the diagram below. Fault Domains reduce the probability that a localized hardware failure will interrupt the service of an application.



Replicas: In addition, Azure embeds high availability features into its other services. For example, Azure Storage maintains three replicas of data. It also allows the option of geo-replication to store backups in a secondary data center. The Azure Content Delivery Network (CDN) allows caching around the world for both redundancy and scalability. Azure SQL Database maintains multiple replicas as well (depicted in the below diagram).



Availability Sets: There are also specific availability features for Azure Virtual Machines that use an Infrastructure-as-a-Service (IaaS) model. In order to achieve high availability with Virtual Machines, you must use Availability Sets. An Availability Set serves a similar function to Fault and Upgrade Domains. Within an Availability Set, Azure positions the virtual machines in a way that prevents localized hardware faults and maintenance activities from bringing down all of the machines in that group.

Key considerations for planning, implementing and operating in a resilient environment

A resilient architecture doesn't happen by chance. You'll need to build resiliency into every step of your application deployment, from the initial planning, to implementation and through day-to-day operations. Your organization will need to make a business decision as to which applications warrant a truly resilient architecture, because building in resiliency is a trade-off between cost and availability.

Below we'll discuss some of the key considerations for each phase.

PLANNING

Much like building a house, you need to start with detailed, well-thought-out plans that have been developed by experts in the field. This can eliminate problems and unnecessary costs down the road. Planning for resiliency typically will also include key activities such as establishing application availability goals and business requirements, identifying potential failure points, automation and backup strategies.

ESTABLISHING AN AVAILABILITY MODEL AND PLAN:

An availability model for your application identifies the level of availability that is expected for your workload. For example, an application that is built to provide traffic information would have a high level of availability expected during rush hour. Similarly, an online retail application would require high availability during the holiday season or promotional periods. Identifying an availability plan is critical as it will inform many of the decisions you'll make when establishing your service. Most services will take a dependency on a third party service – if only for hosting. It's imperative to understand the SLAs of these dependent services and incorporate them into your availability plan.

DECOMPOSING THE APPLICATION BY WORKLOAD:

Applications are typically composed of multiple workloads. Different workloads can, and often do, have different requirements, different levels of criticality to the business, and different levels of financial consideration associated with them. By decomposing an application into workloads, an organization provides itself with valuable flexibility. A workload-centric approach provides better controls over costs, more flexibility in choosing technologies best suited to the workload, workload-specific approaches to availability and security, and flexibility and agility in adding and deploying new capabilities.

Decomposition also allows you to have different internal SLAs at the workload level. An example of this is a weather application. You can split this into two workloads: daily forecasts and storm alerts. During storm alerts, your target SLA may be higher than your daily forecasts target SLA because there will be greater demand and there may be backlash if your application can't provide the data as expected.

Decomposition by workload allows you to have SLAs tailored to the availability needs of the aggregated workload of the composite service.

DEFINING THE AVAILABILITY OR DESIRED SLA

You may or may not publish a public SLA for your service, but regardless, your architecture should target an availability baseline that you will aspire to meet. Decomposing your application to the workload level allows you to make decisions and implement approaches for availability. Delivering 99.99% uptime for your entire application may be unfeasible, but for an individual workload in an application, it is achievable. Even commercial service providers do not offer 100% SLAs because the complexity and cost to deliver that level of SLA is unfeasible or unprofitable.

Depending on the type of solution you are building, there may be a number of considerations and options for delivering higher availability. Taking advantage of existing services, whether yours or from a third party, can provide significant agility in delivering solutions. While attractive, it is important to truly understand the impacts these dependencies have on the overall SLA for your complete application or service.

Understanding the number of "9s": Availability is typically expressed as a percentage of uptime in a given year and is referred to as the number of "9s." For example, 99.9 represents a service with "three nines" and 99.999 represents a service with "five nines."

One common misconception is related to the number of "9s" a composite service provides. It is often assumed that if a given service is composed of 5 services, each with a promised 99.99% uptime in its SLAs, that the resulting composite service has availability of 99.99%. This is not the case.

The composite SLA is actually a calculation that considers the amount of downtime per year. A service with an SLA of "four 9s" (99.99%) can be offline up to 52.56 minutes. Incorporating five services with a 99.99% SLA into a composite introduces an identified SLA risk of 262.8 minutes or 4.38 hours. This reduces the availability to 99.95% before a single line of code is written!

DEFINING THE BUSINESS' TOLERANCE FOR OUTAGE

Every outage should be a concern for any organization, but every organization has its own tolerance for system outages and expected recovery times. A media company that manages both radio stations and digital billboards may have very low tolerance for outage due to audience and advertiser expectations that their ads be seen 24x7. However, a large durable goods manufacturer may have a much higher tolerance simply due to the long lead times involved and the minimum impact a short application or service outage would have on customers.

Even within an organization, there can be varying degrees of tolerance based on business function. An insurance company may define campaign and underwriting systems as critical with very little outage tolerance, while claims management might be more forgiving in terms of downtime. As with any systems planning, the balance will always be cost to mitigate versus impact to business operations.

IDENTIFYING FAILURE POINTS, FAILURE MODES AND FAILURE EFFECTS

To create a resilient architecture, it's important to understand and document what can cause an outage. Understanding the failure points and failure modes for an application and its related workloads can enable you to make informed, targeted decisions on strategies for building in resiliency and availability.

Failure points are areas where failures may result in a service interruption. An important focus is on design elements that are subject to external change. Examples of failure points include database connections, website connections, configuration files and Registry keys.

Failure modes identify the specific manner in which a failure can occur. Examples of failure modes include:

- A missing configuration file
- Significant traffic exceeding resource capacity
- A database reaching maximum capacity

Failure effects are the consequences of failure on functionality. Examples of failure effects include service not available to users or degraded application performance. By identifying the effects of failure and the frequency at which these types of failures are likely to occur, you can prioritize when and how you address the failure points and failure modes of your application or service.

ESTABLISHING A DATA RESILIENCY APPROACH

While Azure will store multiple copies of the data in your application, the data that is stored is driven by the application, workload and its component services. If the application takes an action that corrupts its application data, the platform stores multiple copies of it. When identifying your failure modes and failure points it's important to recognize areas of the application that could potentially cause data corruption no matter where the bad data comes from.

REMOVING THE HUMAN FACTOR THROUGH AUTOMATION

People make mistakes. Whether it's a developer making a code change that could have unexpected consequences, a DBA accidentally dropping a table in a database or an operations person who makes a change but doesn't document it, there are multiple opportunities for a person to inadvertently make a service less resilient.

To reduce human error, one approach is to reduce the amount of human interaction in the process. Through the introduction of automation, you limit the ability for ad hoc, inadvertent deltas from expected behavior to jeopardize your service.

Start by focusing on automation in the building and deployment of a solution. Automation can make it easy for a development team to test and deploy to multiple environments. Development, test, staging, beta and production can all be deployed consistently through automated builds. The ability to deploy consistently across environments works toward ensuring that what's in production is representative of what's been tested.

An example of automation is scripting. Scripting makes deployment and management consistent, predictable and pays significant dividends for the upfront investment.

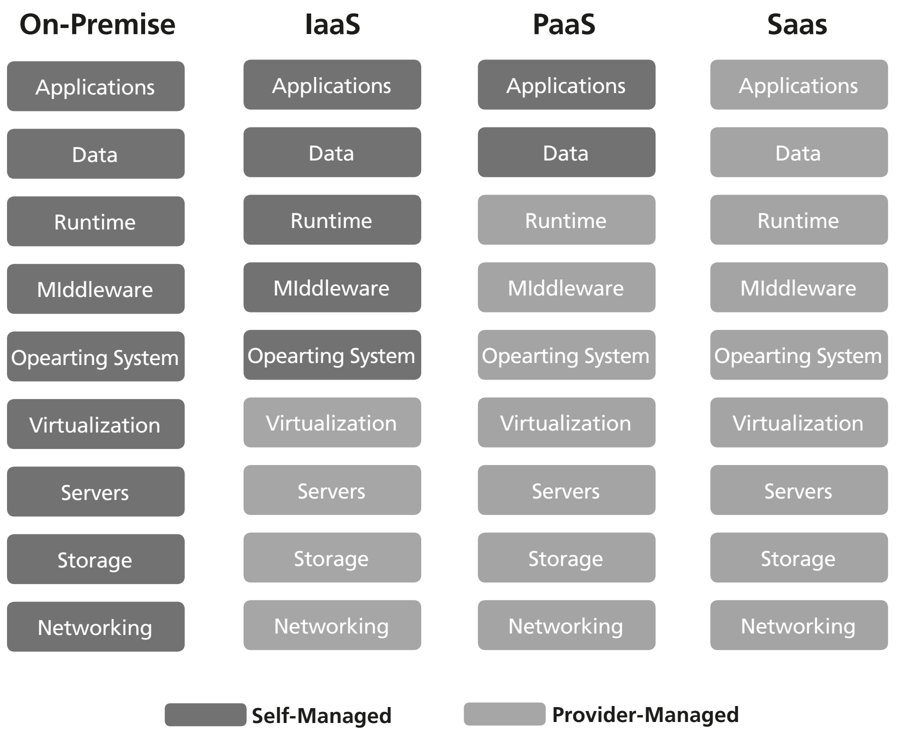
CREATING A BACKUP STRATEGY

Processes for both creating and restoring backup copies of your data store — either in whole or in part – should be part of your resiliency plan. While the concepts of backing up and restoring data are not new, there are new twists to this in the cloud, with options for backup in multiple data centers in multiple geographic locations.

Your backup strategy should be defined with a conscious understanding of the business requirements for restoring data. If a data store is corrupted or taken offline due a disaster scenario, you need to know what type of data must be restored, what volume must be restored, and what pace is required for the business. This will impact your overall availability plan and should drive your backup and restore planning.

Implementation

When implementing your application or service in the cloud, you'll have two choices for how to deploy on Azure – Platform-as-a-Service (PaaS) or Infrastructure-as-a-Service (IaaS). The following diagram shows the varying levels of self-management versus provider management for the different deployment models



PAAS: PLATFORM-AS-A-SERVICE

Cloud platform services, or Platform-as-a-Service (PaaS), are used for application and other development. What developers gain with PaaS is a framework they can build upon to develop or customize applications. PaaS makes the development, testing and deployment of applications quick, simple and cost-effective. With PaaS, your cloud provider manages the operating systems, virtualization, servers, storage, networking and the PaaS software itself. Developers, however, manage the applications.

IAAS: INFRASTRUCTURE-AS-A-SERVICE

Cloud infrastructure services, known as Infrastructure-as-a-Service (IaaS), are self-service models for accessing, monitoring and managing remote data center infrastructures, such as compute (virtualized or bare metal), storage, networking and networking services (e.g. firewalls). Instead of having to purchase hardware outright, users can purchase IaaS based on consumption, similar to electricity or other utility billing.

Compared to PaaS, IaaS users are responsible for managing applications, data, runtime, middleware and operating systems. Providers still manage virtualization, servers, hard drives, storage and networking. Many IaaS providers now offer databases, messaging queues and other services above the virtualization layer as well. What users gain with IaaS is infrastructure on top of which they can install any required platform. Users are responsible for updating these if new versions are released.

IaaS is the most flexible cloud computing model and allows for automated deployment of servers, processing power, storage and networking. IaaS clients have true control over their infrastructure, unlike users of PaaS services.

THINGS TO CONSIDER WHEN CHOOSING PAAS OR IAAS

Reliability: PaaS provides some strong application management tooling, including the ability to detect unstable or misbehaving application instances and take corrective action. However, because of the increase in moving parts involved in a PaaS system, there are more components to fail, and subsequently, increased instances where intervention is required. So, PaaS may be more unreliable than IaaS.

Scalability: In a typical PaaS setup, increasing or decreasing the number of application instances is as simple as issuing a command. This can be particularly useful for predictable high traffic events, or for manually reacting to unexpected bursts in traffic.

IaaS offers little out of the box in terms of instance scale. Assuming your application servers are behind a load balancer, you can add more instances and then deploy the latest version of your application to these instances to achieve the same outcome — though you're probably looking at 15 minutes instead of the 15 seconds for PaaS.

Application Lifecycle Management: If you go the IaaS route, you are responsible for managing applications, from provisioning infrastructure and deploying application changes to scaling application instances. Many quality tools exist in this space, but you're going to miss out on the turnkey access to them that PaaS provides.

PaaS platforms provide a uniform interface to perform common application management tasks. A particular selling point is the ease with which you can replicate entire environments. In simpler setups, you can expect to launch a clone of your entire production environment in a couple of minutes.

IaaS vs PaaS Summary: To decide between IaaS or PaaS, you'll want to consider the application context and the associated long-term business goals. For instance, running an application on bare IaaS is more affordable, but you will need a DevOps team to maintain it. An automated PaaS is a bigger investment, but it can shrink release cycles from weeks to hours and even eliminate some downsides.

PaaS has a definite economic advantage for operations over IaaS for commodity applications where cost of operations breaks the business model. On the other hand, IaaS gives complete control of the OS and application platform stack, which is a requirement for a certain class of applications.

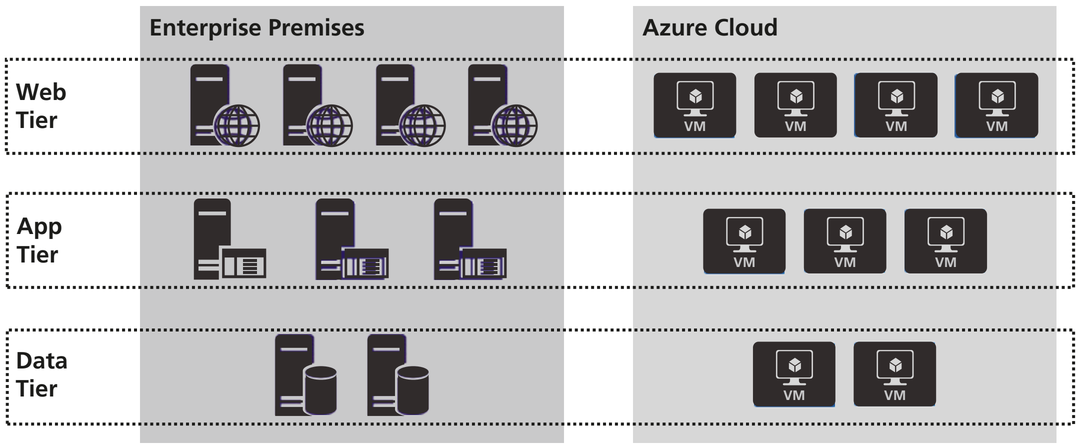
Risk management, agility, cost and nature of the solution are the drivers to consider when deciding between IaaS and PaaS for your architecture solution.

MIGRATING TO AZURE

Migrating from an on-premises environment to the cloud seems simple enough. You take what is working in your existing data center and copy it to the cloud provider's environment, make a few changes for networking and security, and you're in business. In reality, the process is rarely this simple.

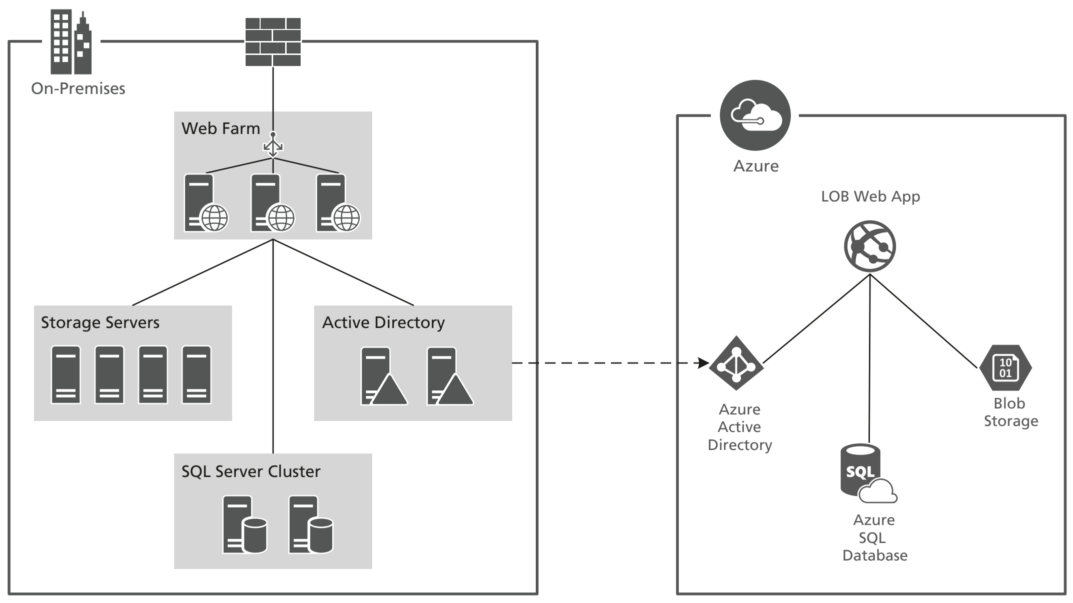
There are two basic categories of approaches when moving to the cloud.

Lift & Shift: The phrase "lift & shift" refers to taking an existing resource (typically a virtualized server) from either an on-premises environment or a hosted private cloud environment and moving it to Azure. See below for a depiction of a typical "lift & shift" scenario.



This means copying the virtualized server files to Azure and configuring them for the Azure environment, including virtual networking and security features. The remainder of the virtualized server can stay largely unchanged. This "lift & shift" approach can be used with either Microsoft platform or Linux servers, assuming that the operating system versions in either case are supported.

Replatforming: The other option is to migrate only the applications and data (and potentially the security processes). This is the traditional PaaS approach and is often appropriate if the organization is planning to upgrade or create a new version of the application anyway. See below for a depiction of a typical replatforming scenario.



Depending on the development approach, replatforming in Azure can be as simple as changing the deployment process and redeploying into the Azure environment. Assuming modern development and testing practices, the only potentially difficult task would be handling security (in the case of Active Directory).

Operation

To build a truly resilient architecture, it should be proactively designed for operations from the beginning. In many cases, operations may not be planned until further along in the lifecycle. Designing for operations typically will include key activities such as establishing a health model, capturing telemetry information, incorporating health monitoring services and workflows, and making this data actionable by both operations and developers.

ESTABLISH A HEALTH MODEL

Development teams often overlook and sometimes completely ignore application health. As a result, services often go into production with two known states: up or down. Designers of resilient services should develop health models that define application health criteria, diminished health state, failure and health dependencies.

Proactively developing a health model outlines failure modes and points, requiring that developers identify and study what-if scenarios in application design phases. To operationalize a health model, a service must be able to communicate its health. It must have a programmatic way to broadcast such information, provide an interface for that health status to be queried interactively, provide mechanisms (or hooks into existing mechanisms) through which admins can monitor application health in real-time and establish mechanisms through which admins can — when necessary — deliver corrective "medicine" to return the application to a healthy state.

TELEMETRY CONSIDERATIONS

When identifying what data to collect and how to collect it, it is important to understand the data and what you intend to do with it.

First, determine if the purpose of the data being collected is to inform or initiate an action. The question to ask is, "How quickly should I react?" Will you use the data near real time to potentially initiate an action? Alternatively, will you use the data in a month over month trending report? The answer to these questions will inform the telemetry approach and technologies used in the architecture.

Next, identify the type of questions you intend to apply to the telemetry data you are collecting. Will you use the telemetry to answer known questions or for exploration? For example, for a business, KPIs (key performance indicators) are answers to known questions. However, a manufacturer who wants to explore device data for patterns that result in faults would be venturing into the unknown. For the manufacturer, the faults are derived from one or more items in the system. The manufacturer is doing exploration and would require additional data.

When you use telemetry to gain insight, you must consider the time required to gain that insight. In some cases, you will leverage telemetry to detect a spike in a device sensor reading that has a window of seconds or minutes. In other cases, you may use telemetry to identify week over week user growth for a website that has a much longer window.

Consider the amount of data you can gather from a signal source within the timeframe to gain insight. You must know the amount of the source signal you need. You can then determine the best way to partition that signal and establish the appropriate mix of local and global computation.

Another consideration is how to record the sequence of events in your telemetry. Many organizations will default to time stamps. Time stamps, however, can be a challenge because server clocks in and across data centers are inconsistent. While time may be synchronized periodically, there is documented evidence that server clocks drift (slowly become more inaccurate). This drift results in changes that may impair effective analysis. For scenarios that require precision, consider alternate solutions, such as leveraging a vector clock implementation.

VISUALIZE TELEMETRY FOR OPERATIONS

Visualizing high-level operation status and lower-level telemetry data is important for the operations staff. Automated notifications will likely be in place based on telemetry data. However, operations will want a dashboard that helps visualize current and historical service performance.

For applications of significant scale, this information can help identify a current issue quickly or predict a future issue. It can also help operations identify the potential impacts and root causes.

Telemetry and reporting are particularly helpful in cases where operations can remediate the errors without code modifications to the services themselves. Examples of activities that operations performs can include deploying more roles and recycling instances.

You can leverage the visualization of historical and real-time telemetry data for:

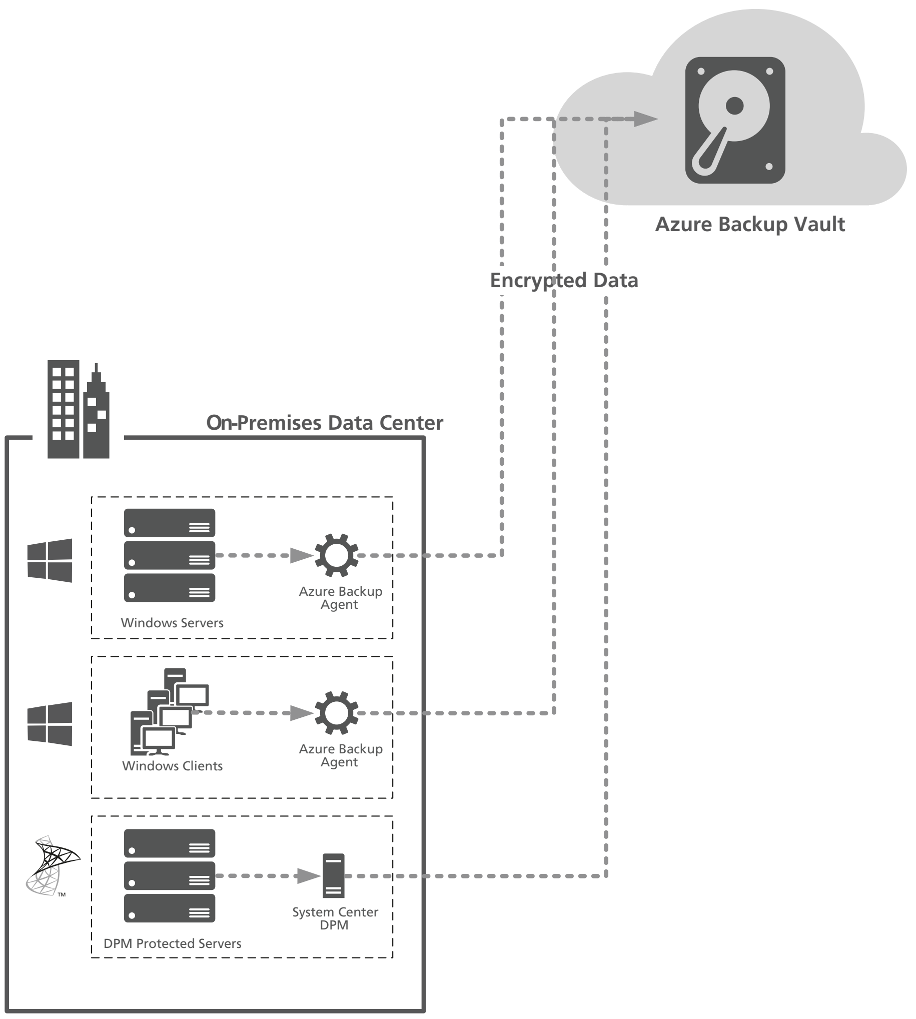
- Troubleshooting
- Post-mortems done for live site issues
- Training new operations staff

Scenarios

The following are real world scenarios showing some types of resiliency available with Azure and how you might use them.

CLOUD BACKUP: ON-PREMISES TO AZURE

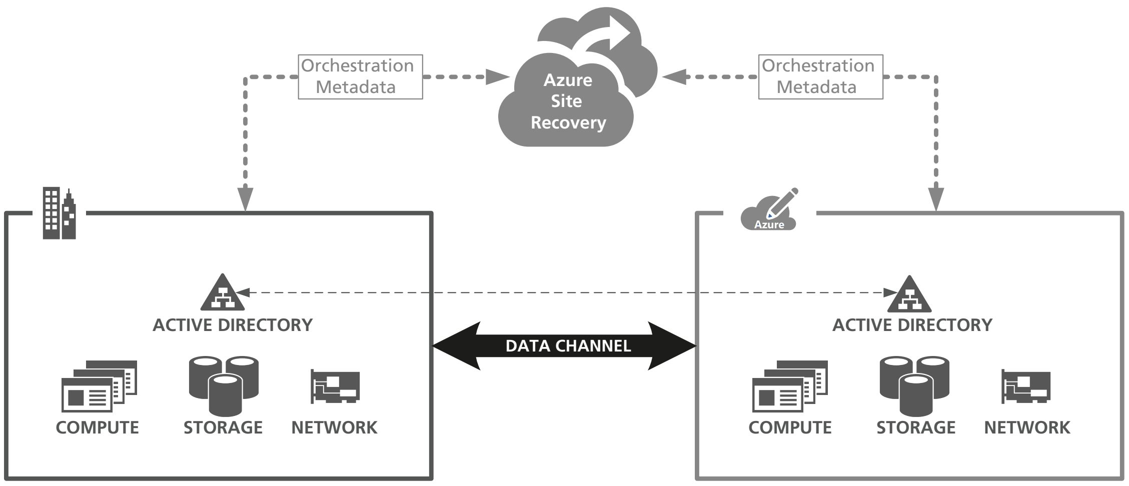
Azure Backup can be used for standard Windows Server configurations, Windows client machines and servers protected by System Center Data Protection Manager (DPM), as depicted in the diagram below. Azure backup stores redundant copies of your data across different Azure data centers. The data is encrypted and transmitted securely to the Azure data center and can be retained in Azure for up to 99 years. Additionally, you manage the encryption keys for your on-premises environments.



If you're not currently using System Center DPM, you can configure and manage a System Center DPM server in Azure itself. This allows on-premises systems to enjoy the advantages of DPM without the complexity of managing an on-premises instance.

TRADITIONAL DISASTER RECOVERY (DR): ON-PREMISES TO AZURE

A DR-to-the-cloud deployment that leverages the Azure platform helps make the restoration of your business operations possible by replicating both the enterprise application environment and the data hosted by the applications running on-premises. Unlike a backup model, where data must be restored to the original application environment, DR-to-the-cloud brings replicated data up at a service provider site. That data then temporarily runs on a replicated application environment at the target site to help ensure complete business continuance until service onsite can be restored (see diagram below).



Technologies like Azure Site Recovery (ASR) can enable powerful and flexible DR solutions that span both physical and virtual machines, automating replication and failover to and between on-premises and Azure data centers. ASR allows you to replicate a variety of workloads including Active Directory, web applications, ERP and CRM solutions, and more. Rackspace engineers will help design the appropriate DR plan based on recovery time objectives (RTO) and recovery point objectives (RPO).

BRIDGE THE GAP: PRIVATE CLOUD HYBRID – BURST TO AZURE

Building a hybrid environment using Azure provides the capability to augment on-premises or hosted private cloud environments as required. This augmentation is usually temporary and is called "bursting."

Bursting capabilities are typically used in situations where high application usage occurs on an infrequent basis. One example of this type of usage is online shopping during the holiday season. Another is product announcements or other news that might drive additional traffic to company web sites.

Through the use of Azure features such as auto-scaling web applications and Azure SQL Data Sync facilities, additional traffic can be load-balanced into the Azure data centers as needed and then scaled back once demand has decreased.

This ability to respond to high usage is one more way Azure can be utilized to create a resilient architecture.

Conclusion

A highly available, resilient application absorbs fluctuations in availability, load and temporary failures in the dependent services and hardware. Planning for and implementing resiliency concepts permit the application to continue to operate at an acceptable user and systemic response level as defined by business requirements or application service level agreements.

Designing, implementing and operating a resilient architecture can be complex. You need the appropriate knowledge and expertise at each step to achieve application availability goals and meet business requirements.

When hardware or software fails in the cloud, the techniques and strategies for managing them are different than when failure occurs in on-premises systems. Azure detects and handles many failures, but there are many types of failures that require application-specific strategies. You must actively prepare for and manage the failures of applications, services and data.

Once you deploy your application, you cannot stop there. You must regularly analyze, test and continually monitor your application portfolio, business needs and the technologies available to you. Azure provides both new capabilities and new challenges to creating robust applications that withstand failures.

Resources

Rackspace downtime cost calculator:
http://www.rackspace.com/en-us/disaster-recovery-planning

Why Rackspace for Microsoft workloads?

Rackspace is the leader in the Gartner Magic Quadrant for Cloud Enabled Managed Hosting, an accolade we received from delivering the world's best service on the world's leading technologies for over 15 years. We have teams of Microsoft Certified Professionals that can help you architect, design, deploy and manage your applications, whether they are on dedicated servers, private clouds, SaaS platforms like Office 365, or even Microsoft Azure itself.

We offer support on a number of technology stacks and applications, including the following:

- Microsoft Azure
- Office 365
- Microsoft Cloud Platform on System Center and Windows Azure Pack
- Microsoft SQL Server
- Microsoft Exchange
- Microsoft Windows Server
- Microsoft SharePoint
- Skype for Business

WHY RACKSPACE?

- *A leader in the Gartner Magic Quadrant for Cloud-Enabled Managed Hosting, North America and Europe 2014 and 2015*
- *Hosting provider to 69% of the Fortune 100*
- *Extensive Microsoft Expertise:*
  - *o Five-time Microsoft Hosting Partner of the Year – more than any other partner*
  - *o Microsoft Gold Certified Partner*
  - *o Gold Partner Microsoft Cloud OS Network*
  - *o 200+ Microsoft certifications, including MCITPs, MCSAs, MCSEs and MCTSs*
  - *o Industry-Leading Exchange Provider*
  - *o #1 SharePoint Hosting Provider (Outside of Microsoft)*
  - *o Six SharePoint MVPs on Staff*
  - *o 85% of SharePoint Hosting Licenses run on our servers*
  - *o Redmond Reader's Choice for Best Hosted Exchange Provider*

About Rackspace

Rackspace (NYSE: RAX), the #1 managed cloud company, helps businesses tap the power of cloud computing without the challenge and expense of managing complex IT infrastructure and application platforms on their own. Rackspace engineers deliver specialized expertise on top of leading technologies developed by OpenStack®, Microsoft®, VMware® and others, through a results-obsessed service known as **Fanatical Support®**.

GLOBAL OFFICES

Headquarters Rackspace, Inc.
1 Fanatical Place | Windcrest, Texas 78218 | 1-800-961-2888 | Intl: +1 210 312 4700
www.**rackspace**.com

UK Office	Benelux Office	Hong Kong Office	Australia Office
Rackspace Ltd.	Rackspace Benelux B.V.	9/F, Cambridge House, Taikoo Place	Rackspace Hosting Australia PTY LTD
5 Millington Road	Teleportboulevard 110	979 King's Road,	Level 1
Hyde Park Hayes	1043 EJ Amsterdam	Quarry Bay, Hong Kong	37 Pitt Street
Middlesex, UB3 4AZ	Phone: 00800 8899 00 33	Sales: +852 3752 6488	Sydney, NSW 2000
Phone: 0800-988-0100	Intl: +31 (0)20 753 32 01	Support +852 3752 6464	Australia
Intl: +44 (0)20 8734 2600	www.rackspace.nl	www.rackspace.com.hk	
www.rackspace.co.uk			

Modified: 11182015