Microsoft

Azure: Designing Resilient & Available Solutions

Michael Royster
Cloud Solution Architect, Microsoft Limited
mroyster@microsoft.com

Building reliable systems is a **shared responsibility**

Customer application

Customer **app** or **workload**, built on the Azure platform.

Resiliency features

Optional Azure capabilities **a customer enables –** high availability, disaster recovery, and backup.

Resilient foundation

Core capabilities **built into the Azure platform –** how the foundation is designed, operated, and monitored to ensure availability.

Agenda

Introduction

Resilient Foundation

Resiliency Features

Resilient Applications

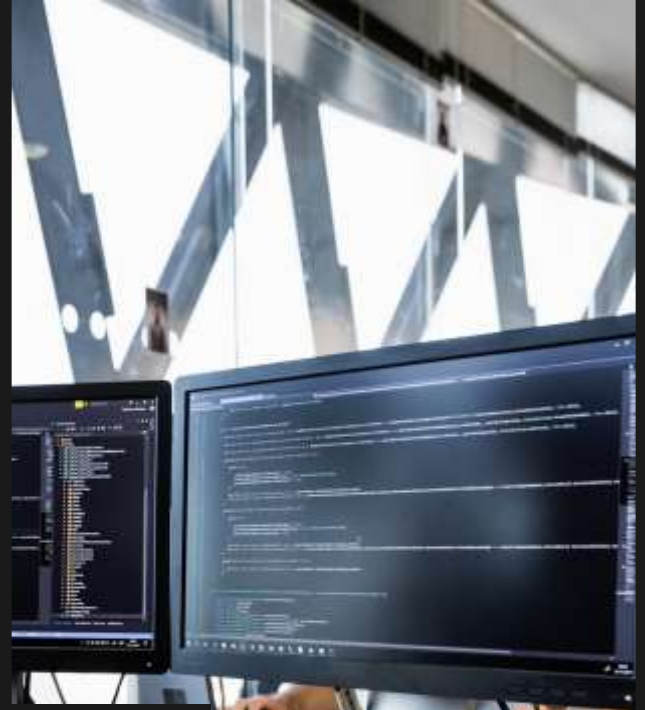
Definitions

Outage is the temporary unavailability of an Azure Service, and can affect some components of the service, such as a data store, or even the entire datacenter. However, after the problem is fixed, the service becomes available again. Typically, an outage does not cause the loss of data. An example of such an outage might be a power failure in the datacenter. Some outages are only short connection losses due to transient or network issues.

Disaster is defined as the permanent, or longer-term loss of a cluster, Azure region, or datacenter. The region or datacenter may or may not become available again, or may be down for hours or days. Examples of such disasters are fire, flooding, or earthquake. A disaster that becomes permanent might cause the loss of some messages, events, or other data.

RTO - Recovery Time Objective The duration of an outage after which the system is expected to have recovered.

RPO - Recovery Point Objective The duration of data loss that is allowable during an outage.



Top cause of outages in the industry? Change.

- Resources created, modified, and deleted 24/7.
- Automated management services introduce infrastructure change as required.
- Workloads scale in / out on demand and at will.
- Service/software updates roll out around the world.
- Constant addition of capacity in every region, balancing various workloads.

Resilient Foundation

99.997%

Azure single-instance VM uptime
(rolling 12-month average, to April 2021)

Resilient foundation: Design

How we design our global fiber network, our evolving datacenters, and storage protections

Design

Global network

Datacenter infrastructure

Storage protection

Operate

Safe deployment

Maintenance & control

ML & failure prediction

Observe

Communications philosophy

Service health & alerts

Scheduled events

Quincy, WA

Azure datacenter design

Microsoft has invested billions of dollars in building a highly secure, scalable, available, and sustainable cloud infrastructure on which customers can rely

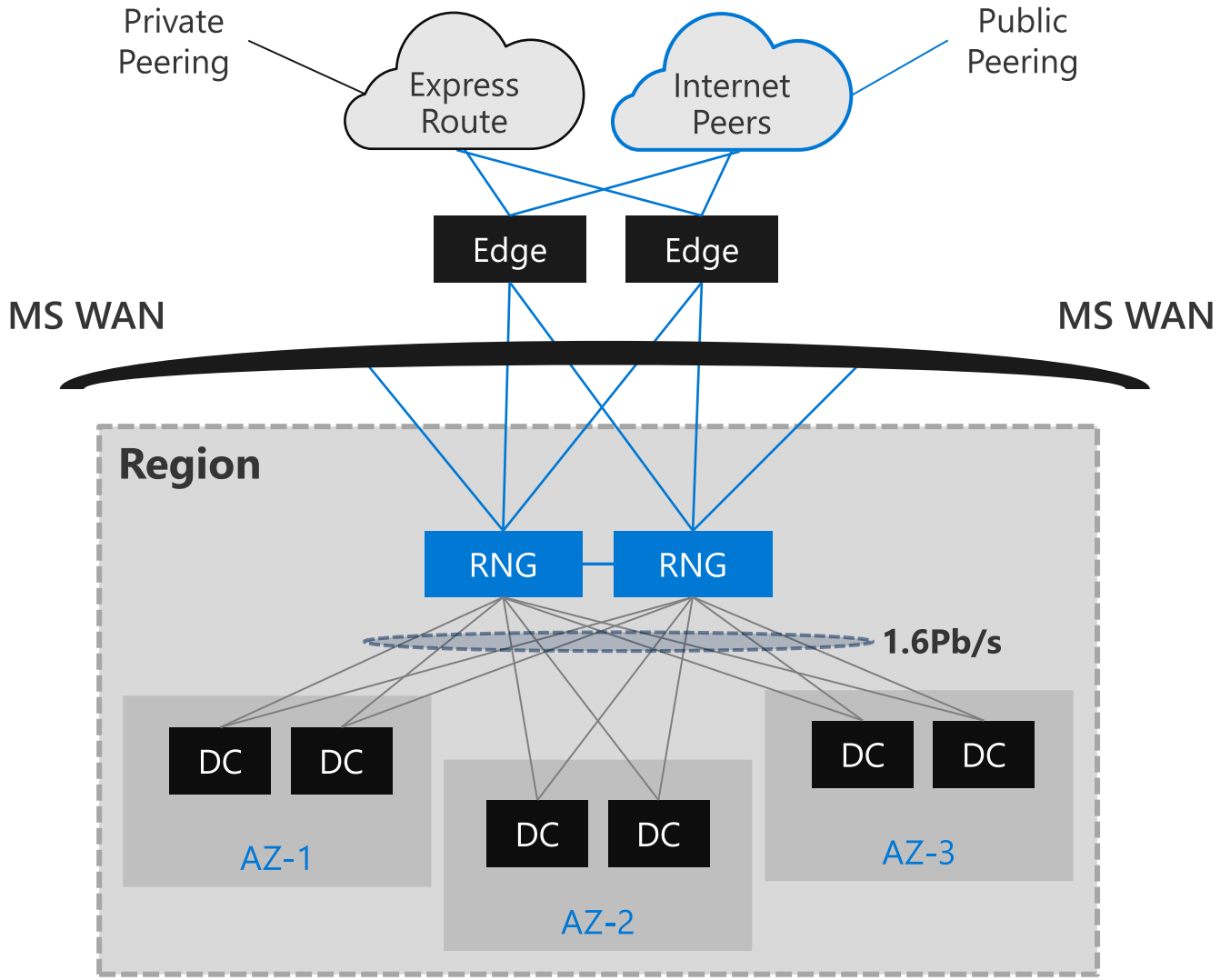
Obvious redundancies — power utility feeds, onsite generators, battery arrays, as well as heating, ventilation and air conditioning (HVAC)

Expanding Availability Zones (AZs) — isolated power, networking, and cooling to provide redundancy against DC-level failures, launched in the 10 largest regions

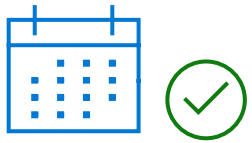
Open Compute Project — sharing hardware designs with the community to learn from feedback, including our datacenter buildings and server specifications

Regional Networking Gateways (RNGs)

Massively parallel, hyperscale datacenter interconnects up to 1.6 Pb/s



Minimizing impact of ongoing maintenance

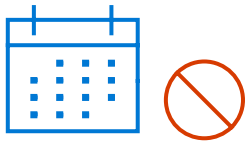


Planned maintenance event (e.g., patching the host OS)

Plan A: Hot-patching – applying changes without incurring downtime

Plan B: Memory preserving – 'pausing' guest VMs (preserving their memory in RAM) while updating the host

Plan C: Self-service maintenance – customer controlled reboots to perform maintenance at the most convenient time (35-day window offered)



Unplanned maintenance event (e.g., hardware predicted to fail)

Trigger: Azure predicts that the hardware or platform is about to fail

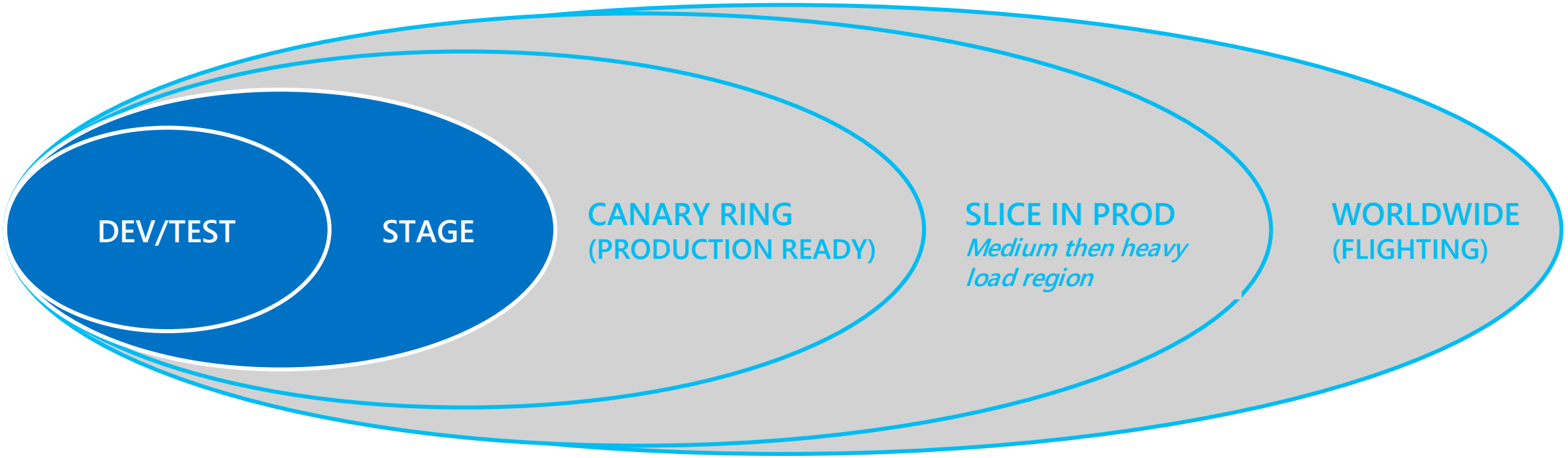
Use Live Migration to evict the node (if possible)

Otherwise, heal the VM into a new node (reboot)

When Live Migration can't complete (e.g., top of rack failure), allow customer to trigger emergency maintenance

Safe Deployment Practices

- Staged rollout through rings of validation **Safe, Compliant and Secure**
- **Canary ring** – subset of Azure customers running workloads in early access environments
- **Worldwide** – changes rolled out with cross region pair rules and health checks



Zone Deployments: All services in Azure deploy one Availability Zone at a time in addition to the region-pair constraint

Resiliency Features

Area	Topic
Application capabilities	**PaaS Application/Compute/Integration resilience**
	Auto Scale in App Services
	High density hosting on Azure App Service using per-app scaling
	Azure Service Fabric Reliable Services
	Auto Scale in API Management
	Service Bus Geo-Disaster Recovery
	Service Bus High Availability
	Geo Distributed Scale with App Service Environments
	Azure Web App Backups
	Deployment Slots in Azure App Service
	IoT Hub High Availability and Disaster Recovery
Compute capabilities	**IaaS resilience**
	Availability Sets
	Availability Zones
	Virtual Machine Scale Sets
	Managed Disks for Virtual Machines in Availability Sets
	Understanding Virtual Machines Reboots Maintance vs downtime
	Designing, building, and operating microservices on Azure
	Azure Site Recovery
	Azure Backup
Storage capabilities	**Storage resilience**
	Azure Storage Replication
	Locally redundant storage (LRS)
	Zone-redundant storage (ZRS)
	Geo-redundant storage (GRS)

Area	Topic
Databases capabilities	**Database Service resilience**
	Cosmos DB High Availability
	Cosmos DB Global Distribution
	SQL Database High Availability
	Active Geo-Replication and Auto-Failover Groups Azure SQL Database
	Automatic SQL Database Backups
	Business Continuity with Azure Database for MySQL
	Backup and Restore in Azure Database for MySQL
	Business Continuity with Azure Database for PostgreSQL
	Backup and restore in Azure Database for PostgreSQL
	Business Continuity with Azure Database for MariaDB
	Backup and Restore in Azure Database for MariaDB
	Redis Clustering for a Premium Azure Redis Cache
Networking capabilities	**Network resilience**
	Azure Load Balancer
	Highly Available Network Virtual Appliances
	Highly Available Cross-Premises and VNet-to-VNet Connectivity
	ExpressRoute
	Disaster Recovery using Azure DNS and Traffic Manager
	Autoscaling, Zone Redundant Application Gateway
	Azure Firewall
	Azure Virtual WAN
	Azure Front Door and Load Balancing
	Protecting DNS Zones and Records
Other capabilities	**Security/Regional/Other resilience**
	Azure Key Vault Disaster Recovery
	Azure Scheduler for High Availability
	Azure Regions
	Availability Paired Regions
	Design for resilience
	Role Based Access
	Azure Monitor
	Azure Monitor and Autoscaling Based on Performance or Schedule
	Azure Advisor High Availability Recommendations
	Azure Service Health
	Azure Policy
	Azure Blueprints

Azure Geographies

Each Azure Geography is comprised of **one or more Azure regions** that sit within a fixed **data residency boundary**

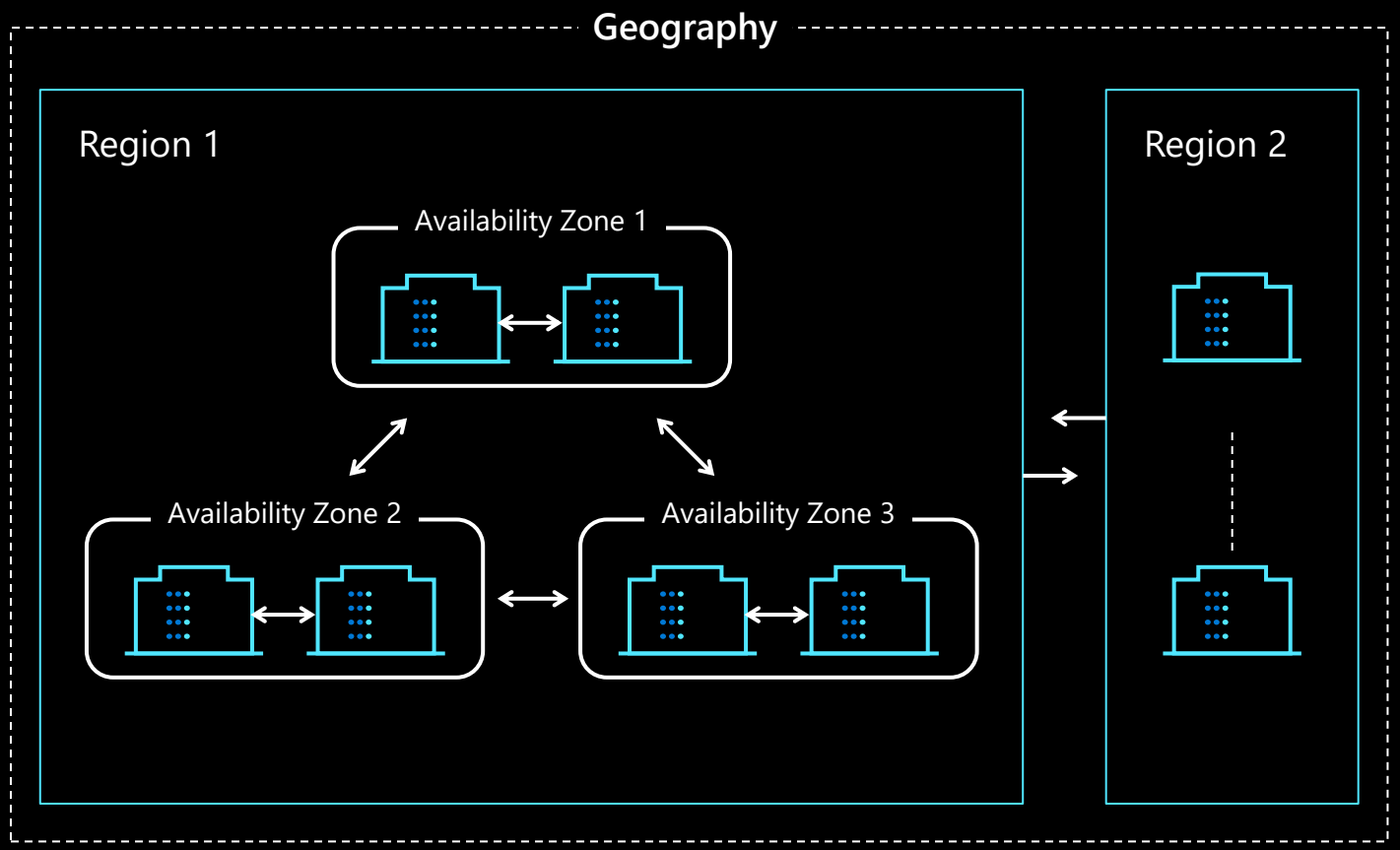
- Usually a single country, to ensure that it can meet government regulations and requirements of restricted industries

In each geography, we have (or will have, by the end of 2021) **at least one region** equipped with **Availability Zones**

- The benefits of this region also include our most significant capacity thresholds, and our broadest service availability

Most geographies **include one or more additional regions,** to enable disaster recovery scenarios in the same data residency boundary

- Where an Azure Geography is not yet large enough to include multiple regions, then DR capabilities are provided through Availability Zones



Azure Regions

Each Azure Region is comprised of **one or more Azure datacenters** that sit within a fixed **metro-level boundary**

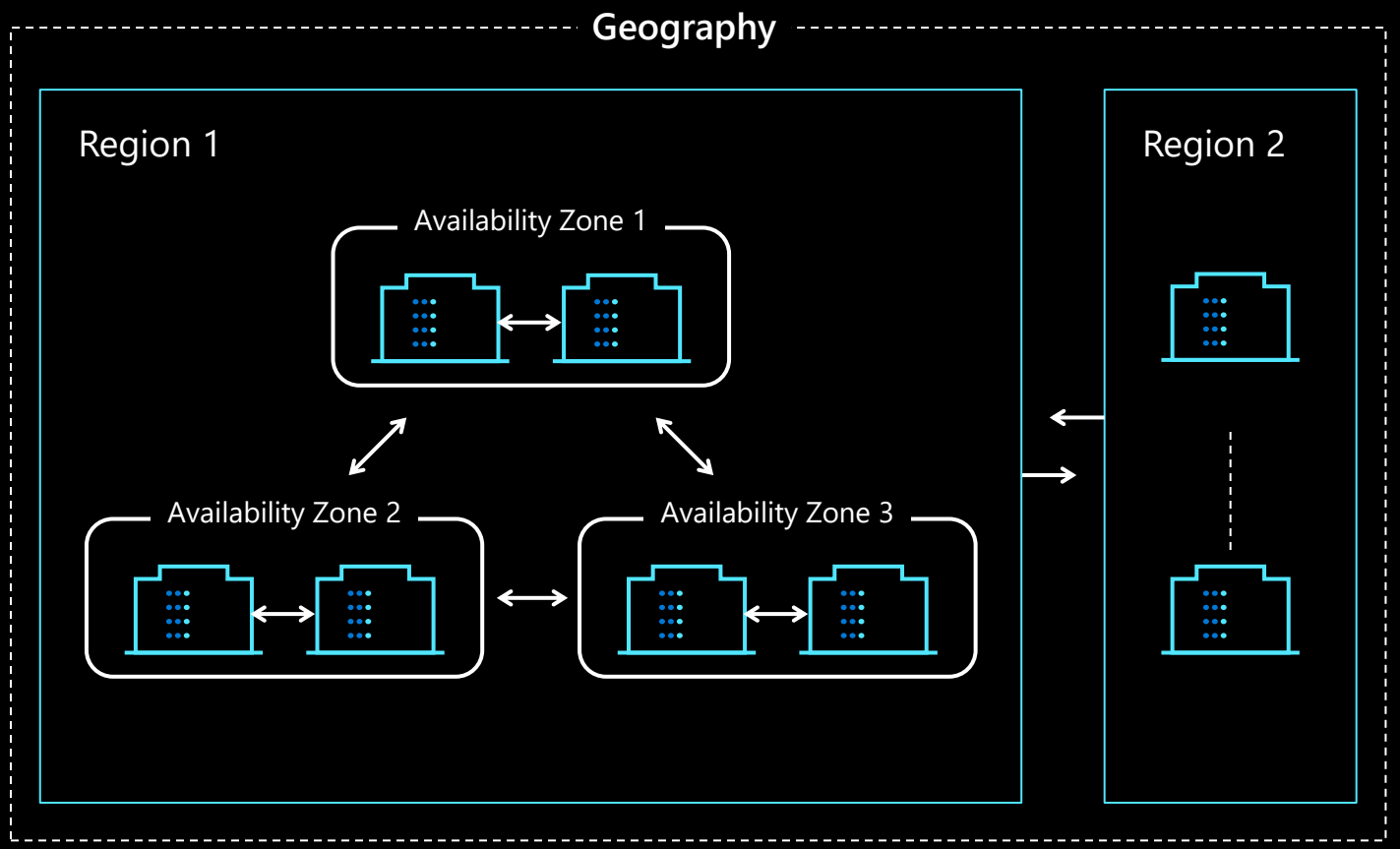
- Usually a single city, within a latency-defined perimeter to enable synchronous data replication

The Azure portal lists some regions as **"preferred"** based on Availability Zone presence/potential and service availability commitments

- Preferred regions generally receive new Azure services first, including specialty services e.g., NDv2 VMs

Azure serializes platform updates across **'paired regions'** to ensure that only one region in each pair updates at a time

- Pairs are fixed for maintenance and storage account replication, but most Azure services let users select their own secondary region for disaster recovery (e.g., planning a failover using Azure Site Recovery)



Azure Availability Zones (AZs)

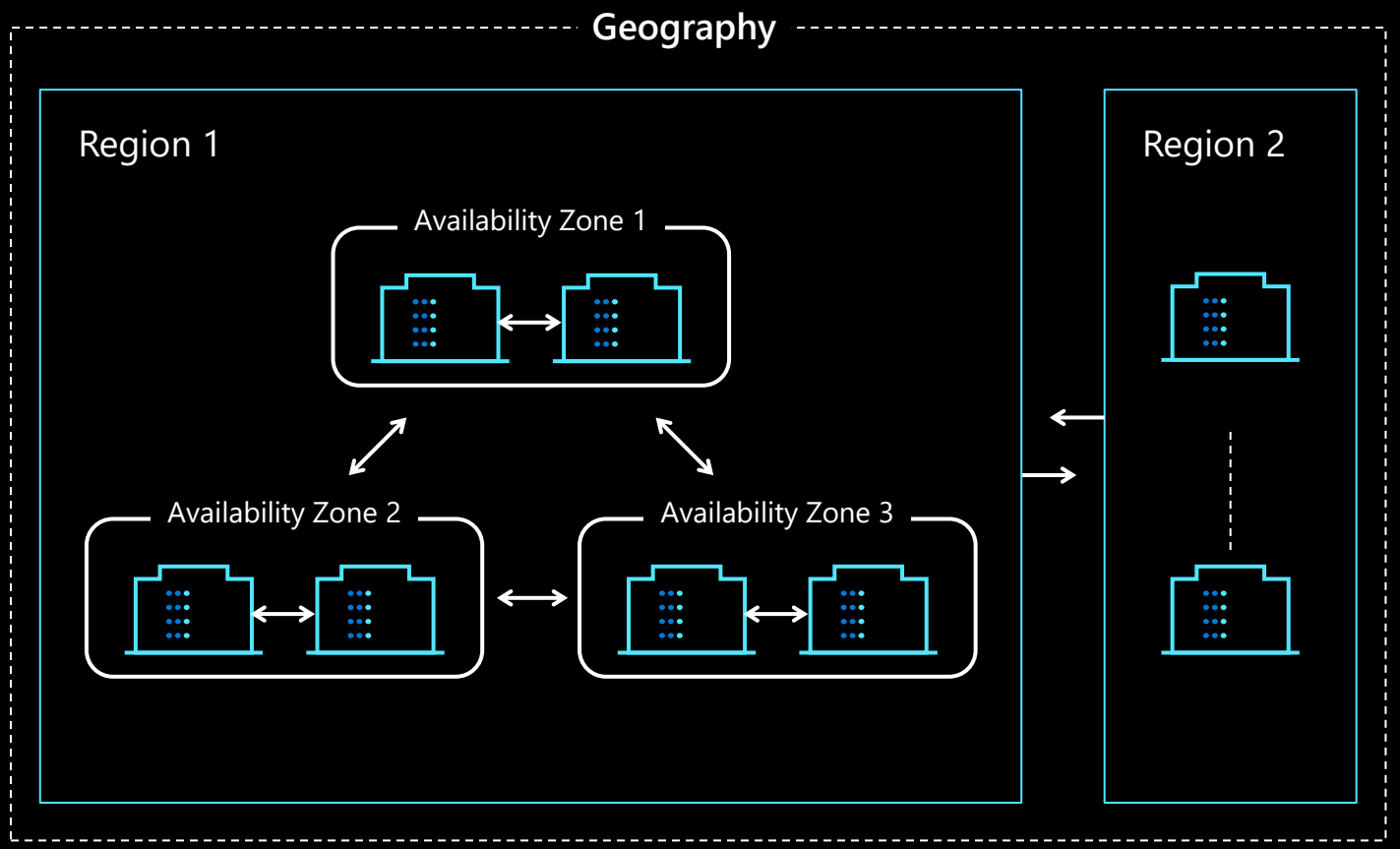
If an Azure region supports AZs, then the region is comprised of **three physically-separated zones** (each comprised of one or more datacenters)

- Each zone has completely independent power, cooling, and networking infrastructure

A facility-level failure (e.g., a datacenter outage) **will affect only one** Availability Zone within the region

- Three zones is the minimum, to support quorum-based workloads (e.g., SQL, Service Fabric, MongoDB)

A **latency-defined perimeter** dictates how far apart Availability Zones can be without compromising networking performance



Latency Between Zones

Region: northeurope
VM Type: Standard_D8s_v3
Latency:

	zone 1	zone 2	zone 3
zone 1		59.9 us	66.8 us
zone 2	60.4 us		52.3 us
zone 3	69.3 us	49.4 us	

Region: westeurope
VM Type: Standard_D8s_v3
Latency:

	zone 1	zone 2	zone 3
zone 1		666 us	54.5 us
zone 2	656 us		657 us
zone 3	54.5 us	655 us	

Importance of Proximity Placement groups for latency sensitive applications, e.g. SAP

SAP-on-Azure-Scripts-and-Utilities/AvZone-Latency-Test.ps1 at main · Azure/SAP-on-Azure-Scripts-and-Utilities (github.com)

Azure storage resiliency solutions

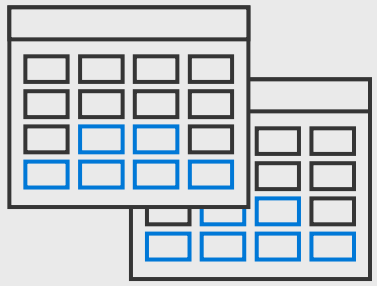
- Azure storage provides replication options based on availability needs

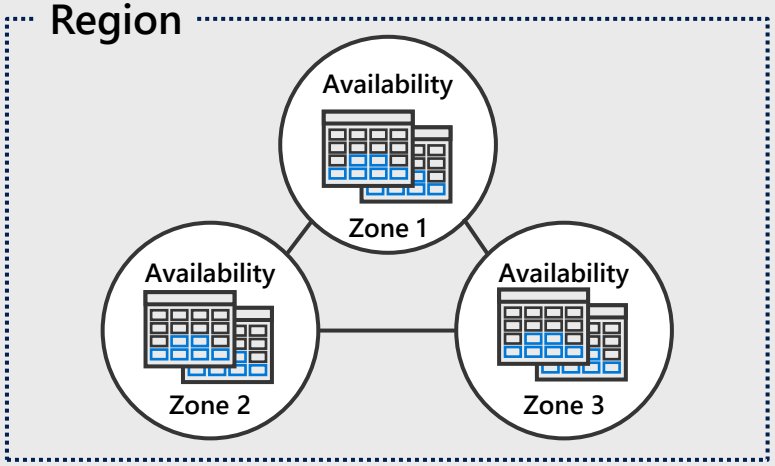
Storage	Local/zone/geo redundant storage

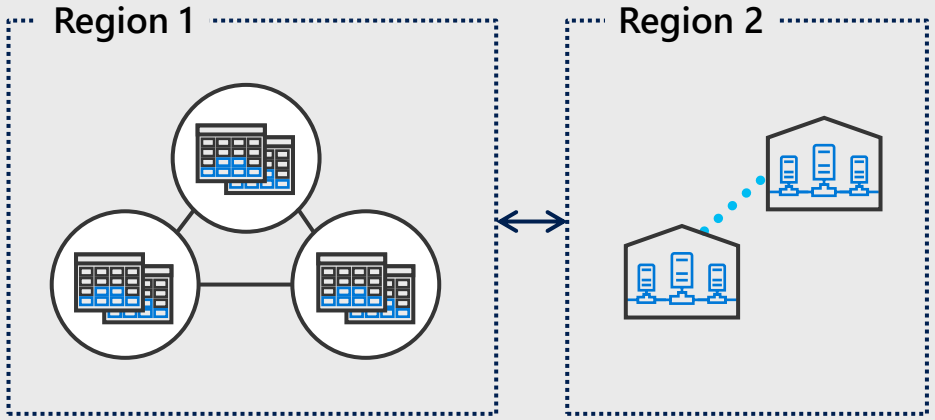
LRS
99.9% (Read availability)

ZRS
99.9% (read availability)

RA-GRS
99.99% (read availability)







Locally redundant storage

The simplest, low-cost replication strategy that Azure Storage offers.

Zone-redundant storage

A simple option for high availability and durability.

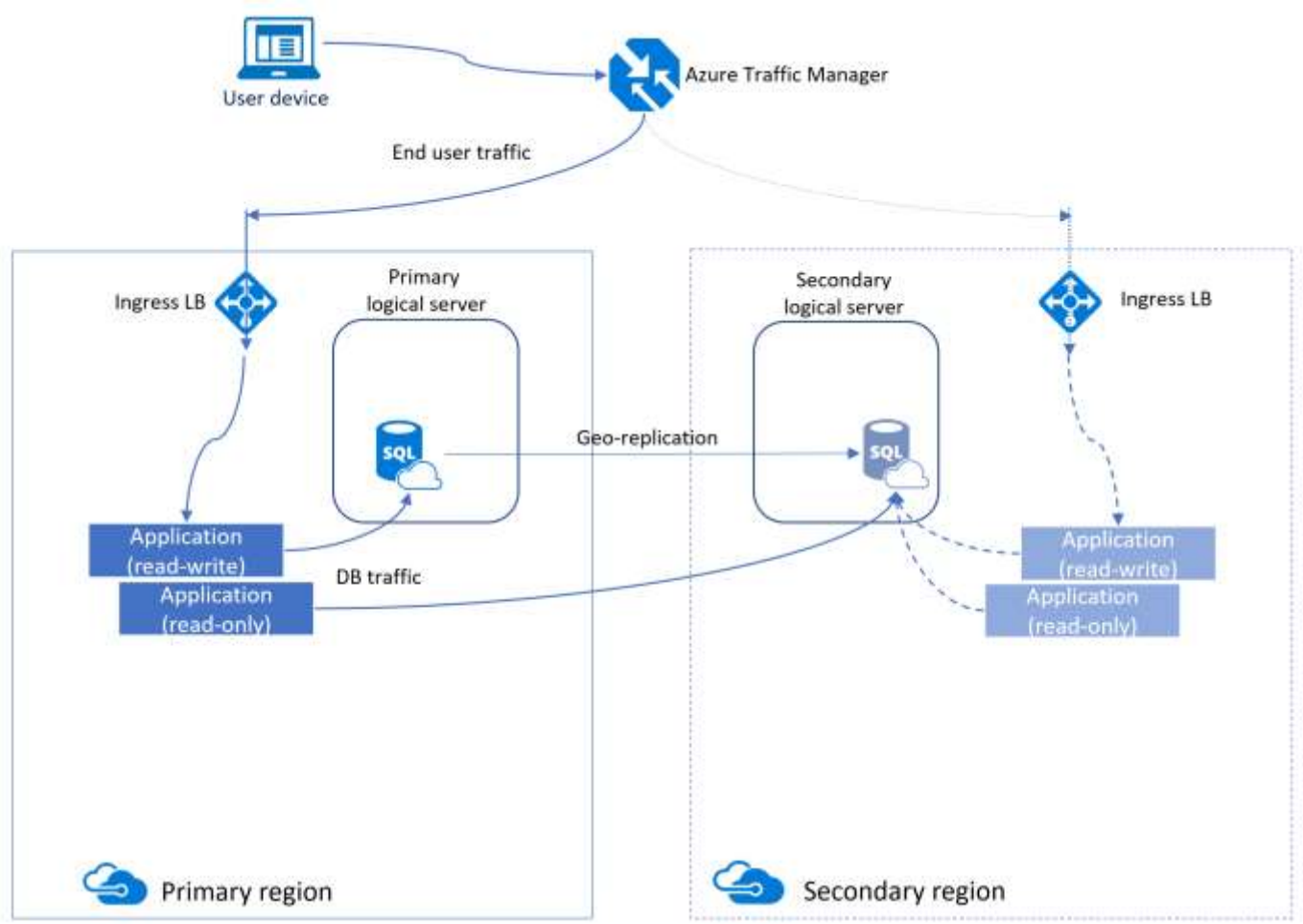
Geo-redundant storage

Cross-regional replication to protect against region-wide unavailability.

Azure SQL Database Business Continuity Features

	Feature	Use Cases
Restorability	Point in Time Restore (PiTR)	• Restore from customer error or to a baseline. • Used for dev/test environments.
	Restore from LTR Backup	• Restore beyond PiTR timeline • Compliance • Auditing
Business Continuity	Geo-Restore	• Cheapest disaster recovery solution • Restore to any Azure region, even when source datacenter is inaccessible. • Low Cost, High RTO (~ 24h)
	Active Geo Replication	• Higher Cost, Minimal RTO • Read-scale • Application Migration / Application upgrades
	Auto Failover Groups	• Medium Cost, Medium RTO • Automated failover • Read Scale

Azure SQL Database Active Geo-replication



CosmosDB Multi-Master

Perfect for Intelligent Cloud

and Intelligent Edge Applications

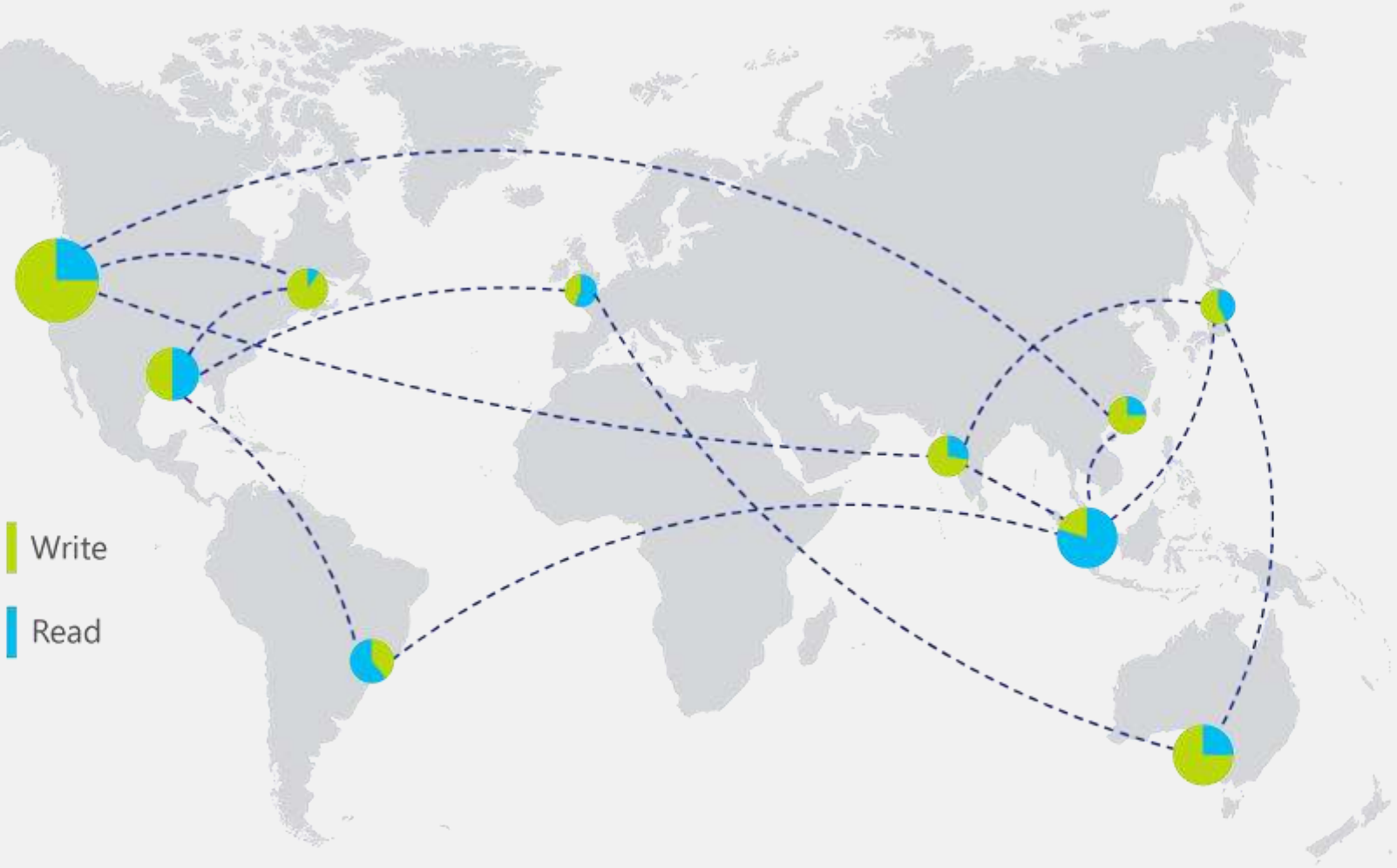
Write scalability around the world

Low latency writes around the world

99.999% High Availability around the world

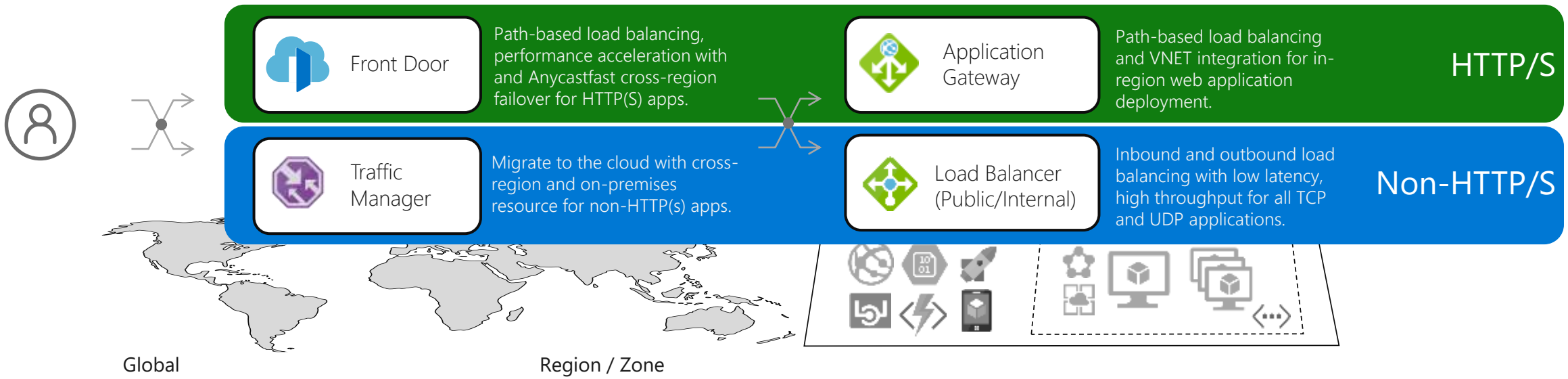
Well-defined consistency models

Comprehensive conflict management



Azure Cosmos DB Multi-Master

Load balancing in Azure



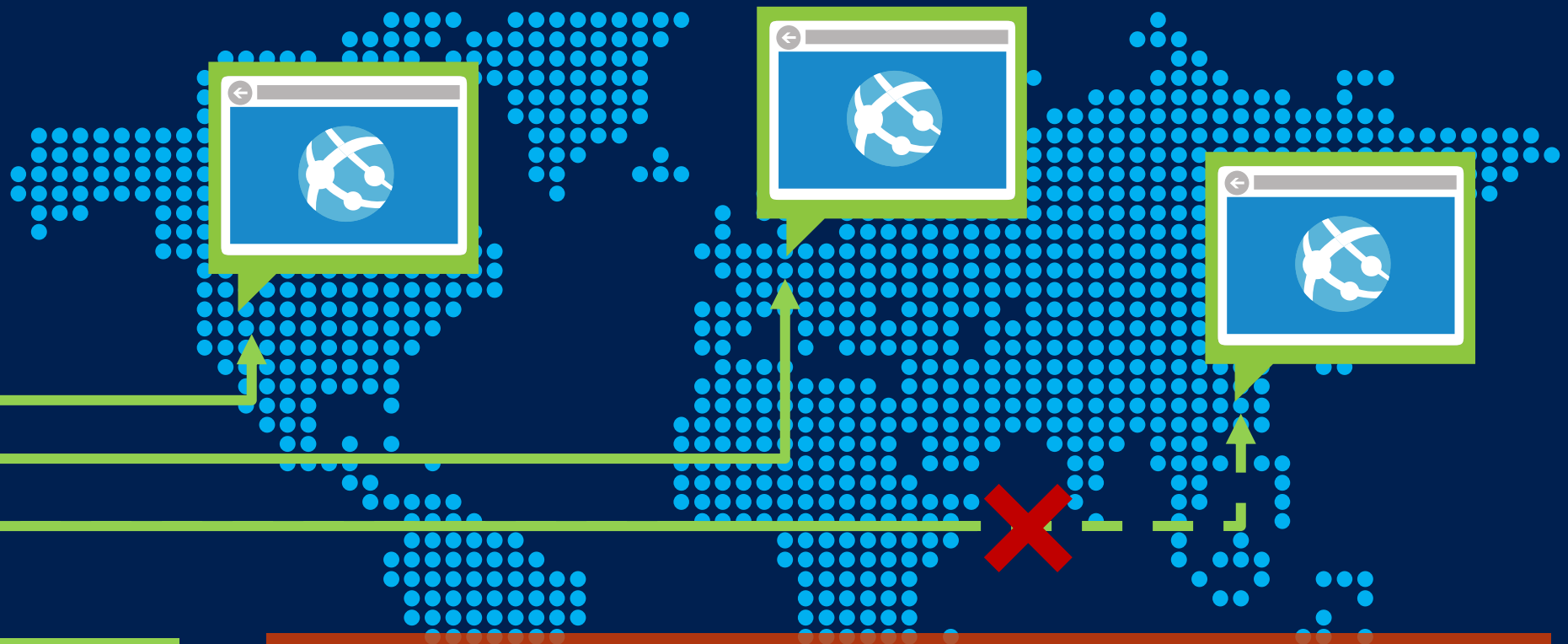
Global

Route to your closest available service region in Azure or your on-premises DC. Load balance across your application scale-units or deployments or endpoints.

Regional

Route across zones and into your VNET. Private IP space routing and between your resources to build your regional application. Load balance across your VMs or containers within a region behind a VNET.

Traffic Manager: Intelligent customer routing



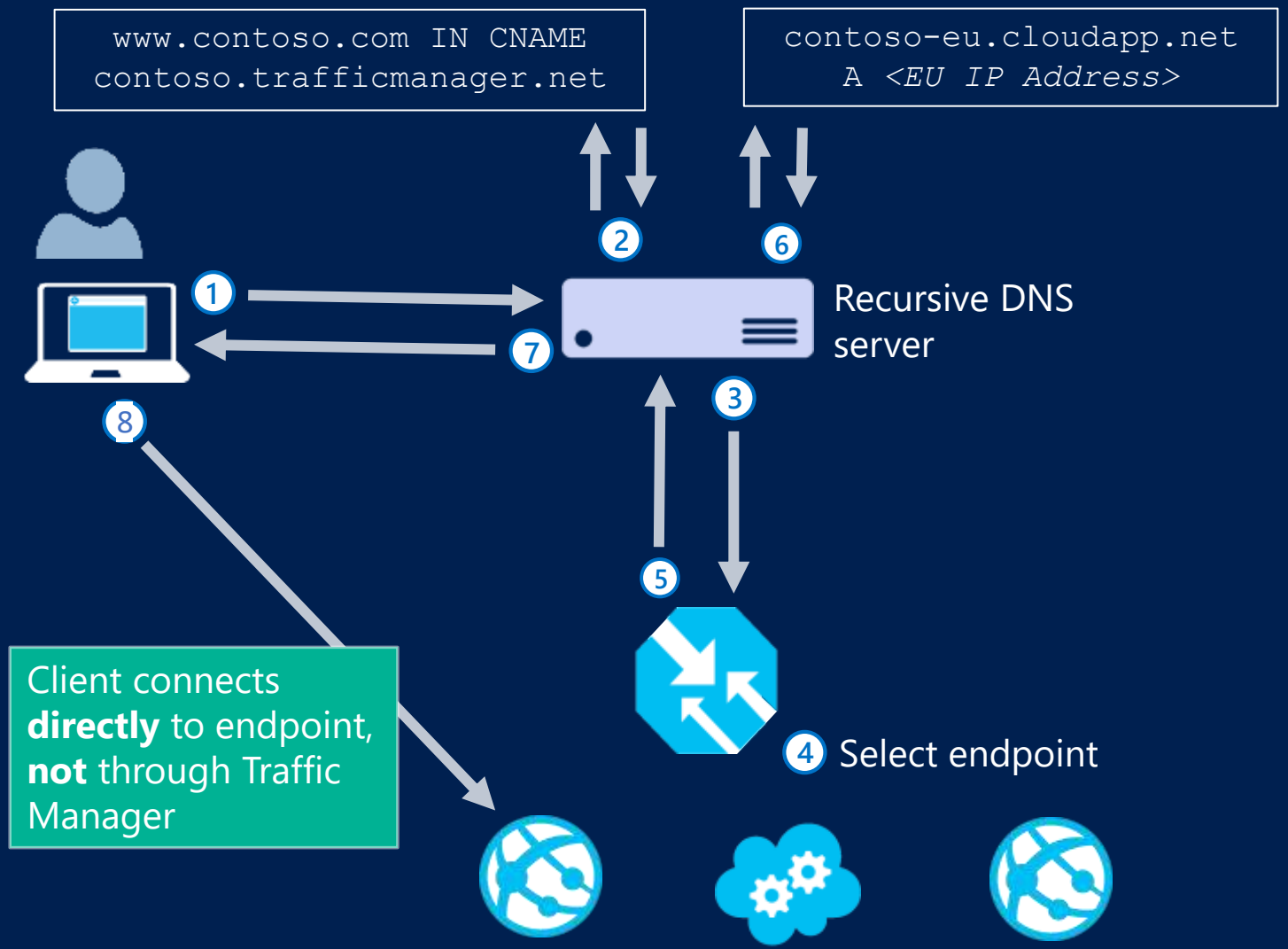
www.yourapp.com

Traffic Manager provides

- **Flexible DNS-based global traffic routing for Azure-based and external services**
- **Automatic endpoint health checks and failover**

How Traffic Manager Works

1. User requests 'www.contoso.com'; client starts with DNS request
2. Recursive DNS service resolves www.contoso.com to contoso.trafficmanager.net
3. Recursive DNS service queries Traffic Manager name server
4. Traffic Manager chooses which endpoint to return, based on **traffic-routing method** and **health checks**
5. Endpoint is returned in DNS
6. Recursive DNS service completes the DNS chain to obtain IP address
7. DNS results returned to client
8. Client connects **directly** to endpoint IP address, **not** through Traffic Manager



Azure Front Door Service

Global secure entry-point to the cloud

- Application acceleration at Microsoft's edge
- Global WAF at edge
- **Global HTTP load balancing with fast failover (anycast)**
- Path based load balancing (microservice arch.)
- Massive SSL offload
- Integrated static content caching
- Global app dashboard, service insights

Global HA, BCDR

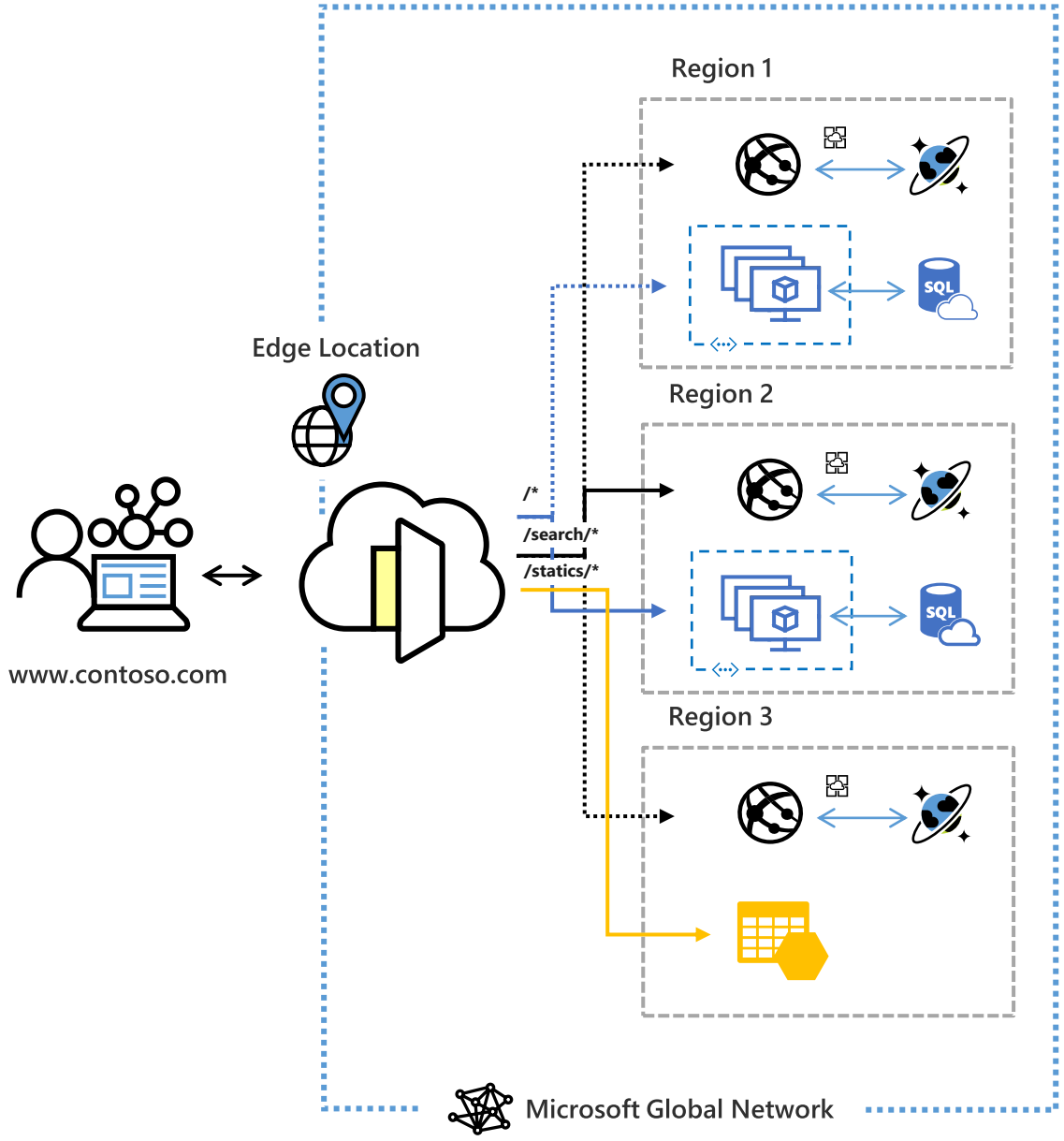
Enable fast-failover for regional services, microservices at the Edge with active path monitoring

Security at the Edge

Stop threats where they come from at the Edge with DDoS protection and customizable WAF

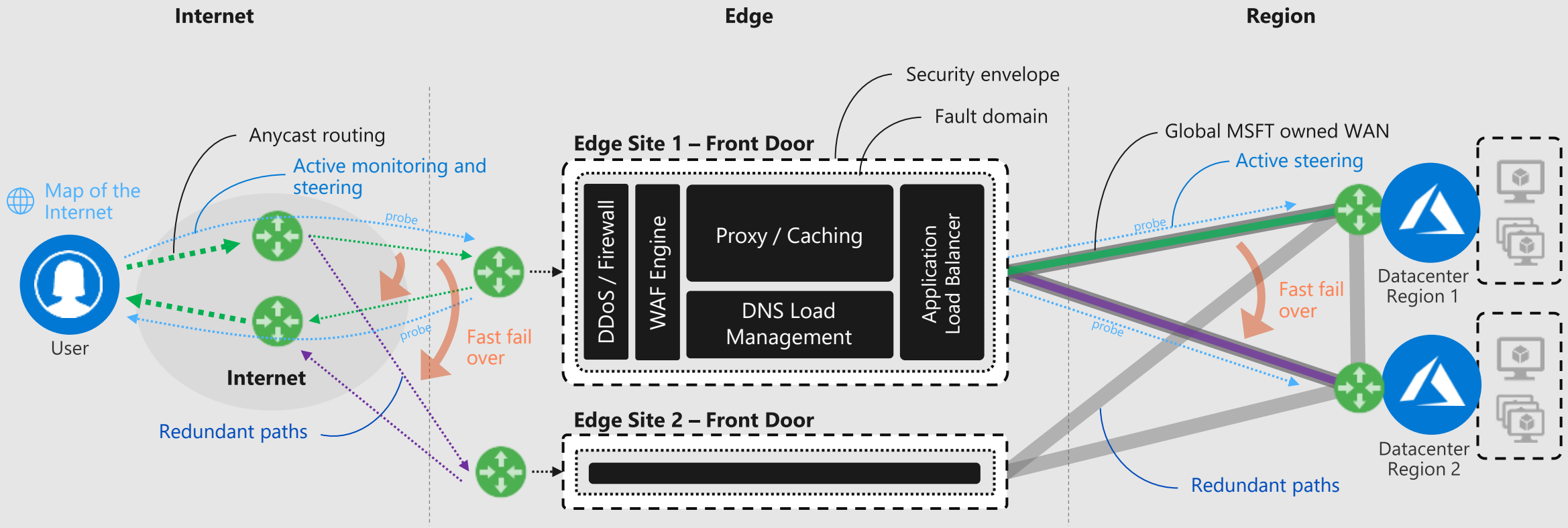
Faster apps

Reduce latency and increase throughput for apps by offloading SSL at the Edge and accelerating requests



Enterprise grade architecture with Front Door

High availability though consistent redundancy, edge-isolated fault-domains and live steering



Customer Application

Application

- Resilient Patterns
- Challenges with state
- All up architectures

Cloud Resilience Architectural Patterns (a few examples)

- Retry
- Bulkhead
- Cache Asside
- Competing Consumer
- Deployment Stamps
- Circuit Breaker
- Throttling

Retry Pattern

Enable an application to handle transient failures when it tries to connect to a service or network resource, by transparently retrying a failed operation.

1: Application invokes operation on hosted service. The request fails, and the service host responds with HTTP response code 500 (internal server error).
2: Application waits for a short interval and tries again. The request still fails with HTTP response code 500.
3: Application waits for a longer interval and tries again. The request succeeds with HTTP response code 200 (OK).

Retry Pattern Idempotent Requests

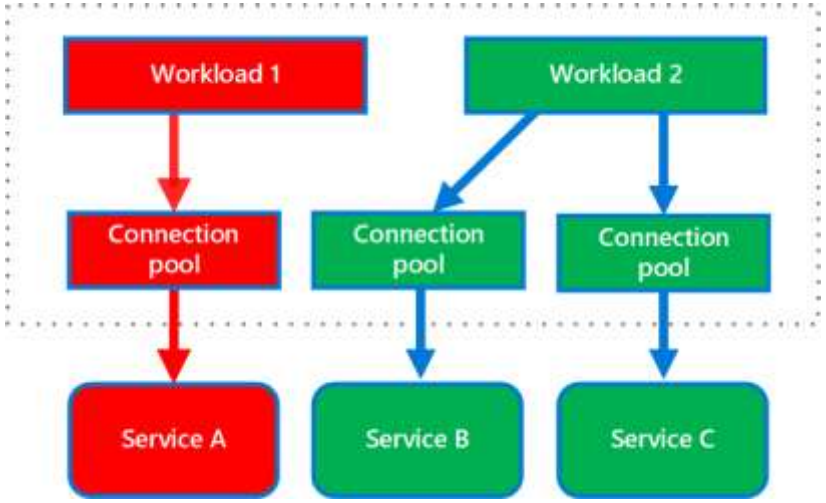
Idempotent is the property of certain operations in mathematics and computer science whereby they can be applied multiple times without changing the result beyond the initial application.

- *Debit £50 from Account 0071234*
- *Debit £50 from Account 0071234 on 16th November 2021 18:01:10.25*

Bulkhead Pattern

The Bulkhead pattern is a type of application design that is tolerant of failure. In a bulkhead architecture, elements of an application are isolated into pools so that if one fails, the others will continue to function.

It's named after the sectioned partitions (bulkheads) of a ship's hull. If the hull of a ship is compromised, only the damaged section fills with water, which prevents the ship from sinking.



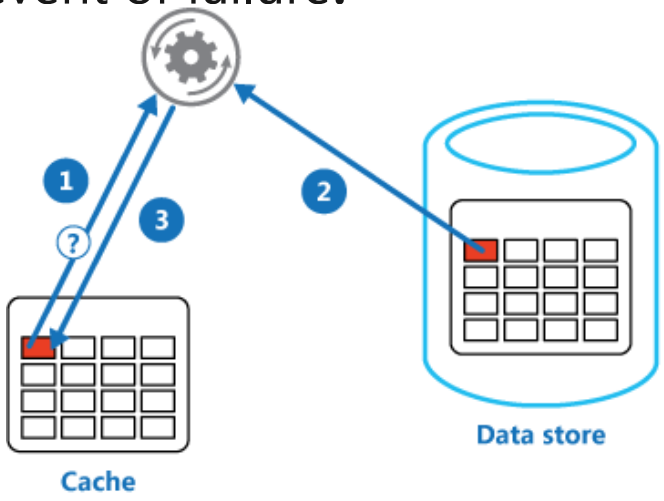
Goal: Graceful degredation

Cache Aside

Load data on demand into a cache from a data store. This can improve performance but also resilience.

Netflix Hystrix examples

- Application Caching: e.g. tiering, last catalogue presented, general customer group, most watched movies
- System Caching cache last service IP addresses to be fault tolerant of DNS failure to use in event of failure.
- Hold customer identifier in URL



1: Determine whether the item is currently held in the cache.
2: If the item is not currently in the cache, read the item from the data store.
3: Store a copy of the item in the cache.

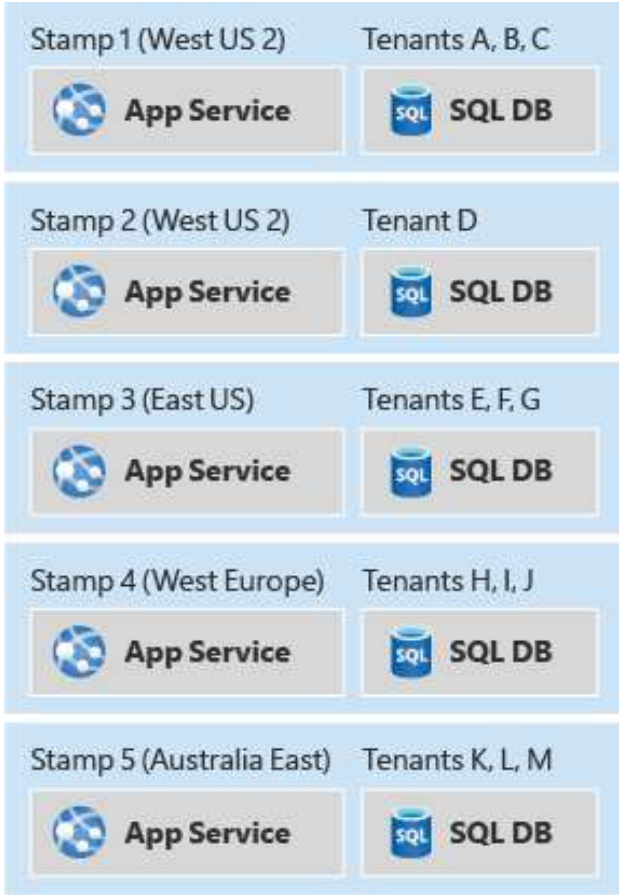
To Cache or not to Cache

- OK to Cache Immutable Data
  - It's Never Wrong
  - Never Have to Shoot Down the Cache!
- Consider Stability of the Data
  - Don't Want to Misunderstand It
- Messages Are Easily Cached
  - They Should Be Immutable and Stable
- Reference Data Should Be Versioned
  - The Version Is Immutable
- Some Data Should Not Be Cached
  - If It Changes, the Cache May Be Out of Sync

Data on the Outside versus Data on the Inside Pat Helland Microsoft Word - i.doc (cidrdb.org)

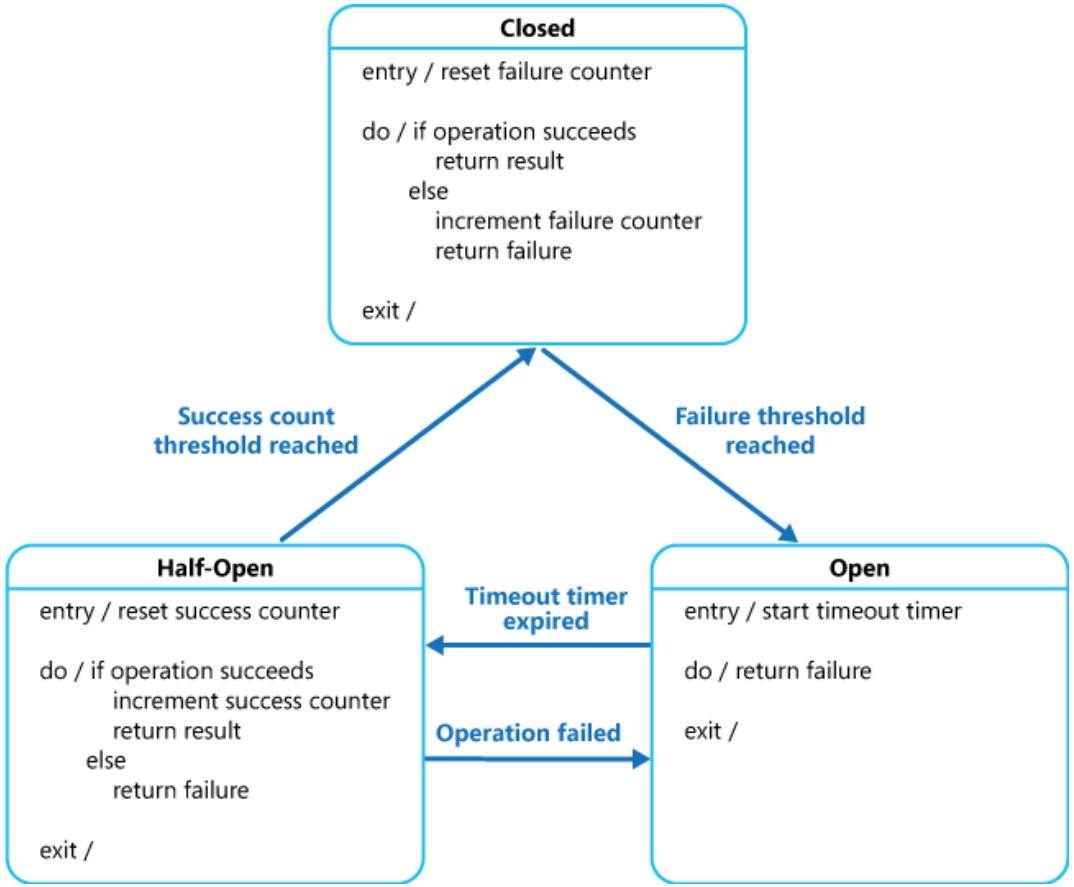
Deployment Stamp Pattern

The deployment stamp pattern involves provisioning, managing, and monitoring a heterogeneous group of resources to host and operate multiple workloads or tenants. Each individual copy is called a stamp, or sometimes a service unit or scale unit.



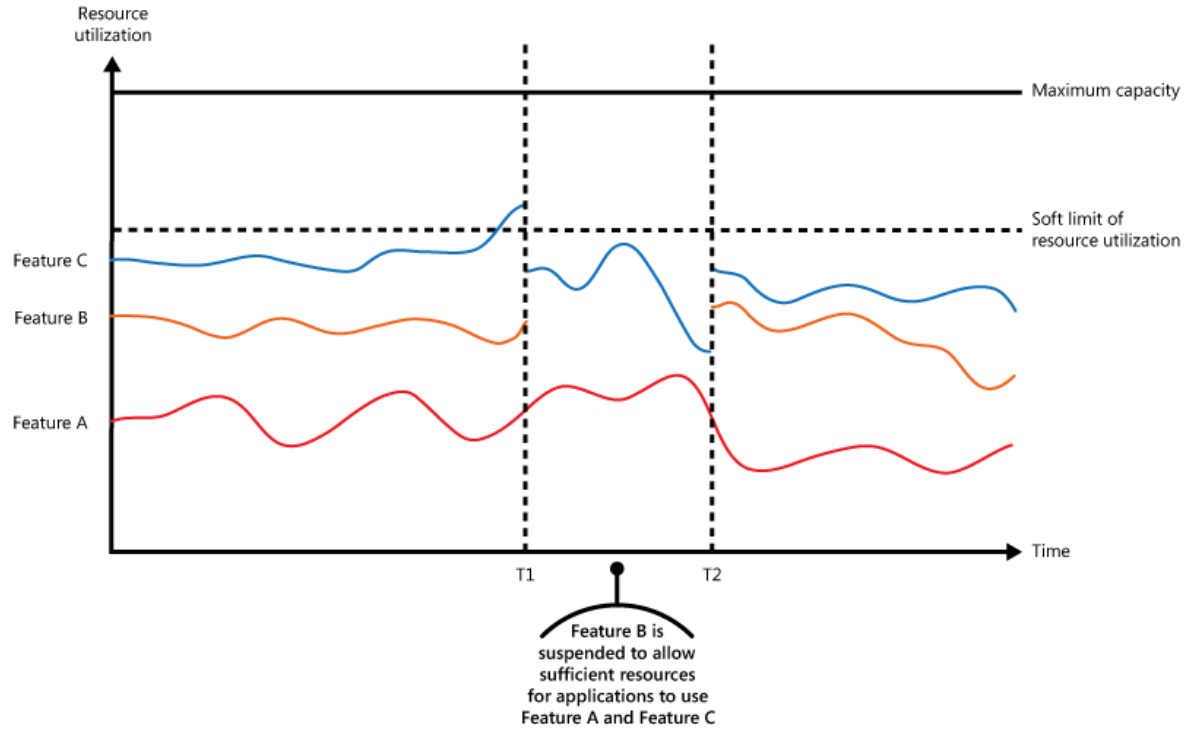
Circuit Breaker Pattern

Handle faults that might take a variable amount of time to recover from, when connecting to a remote service or resource. This can improve the stability and resiliency of an application.

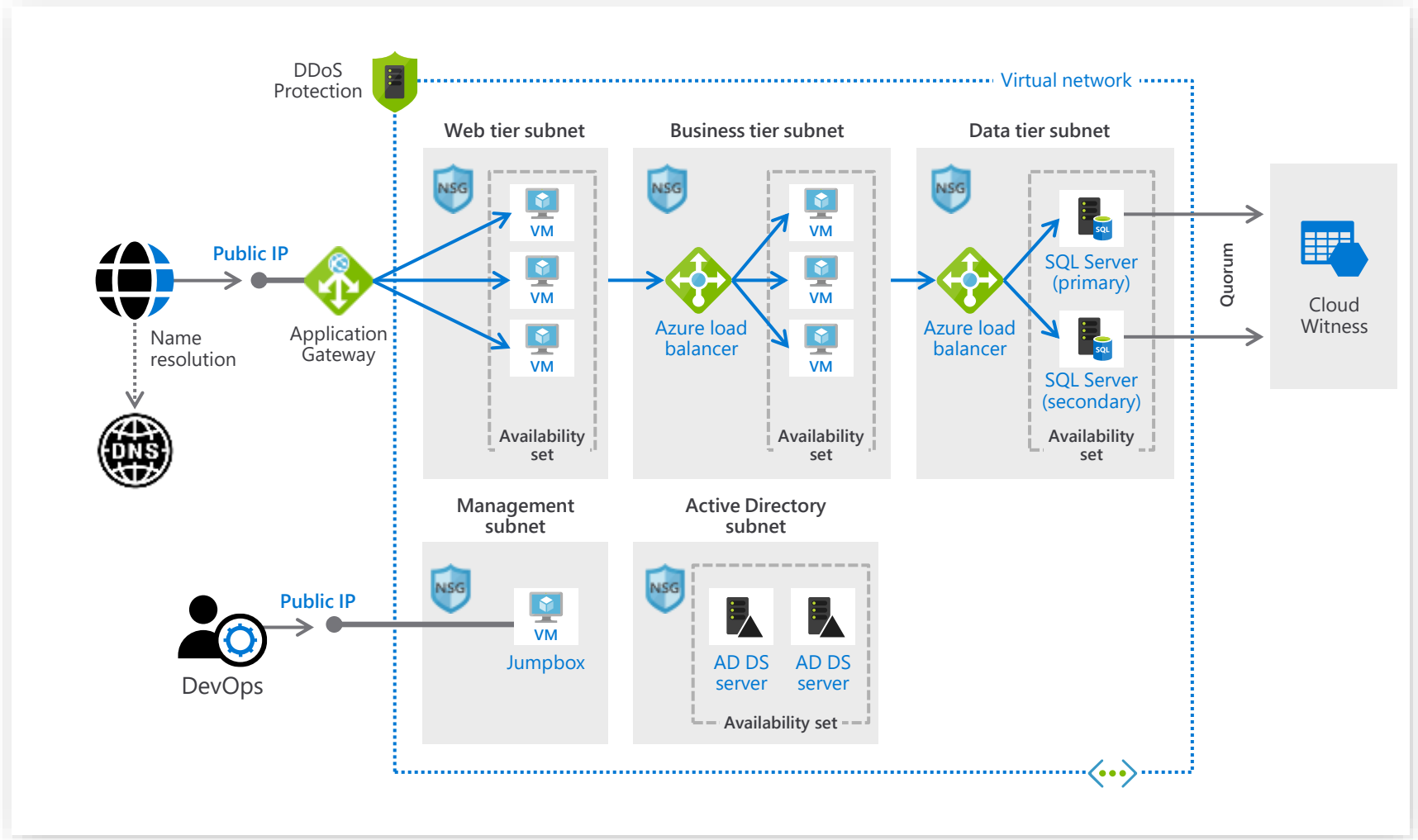


Throttling Pattern

Control the consumption of resources used by an instance of an application, an individual tenant, or an entire service. This can allow the system to continue to function and meet service level agreements, even when an increase in demand places an extreme load on resources.

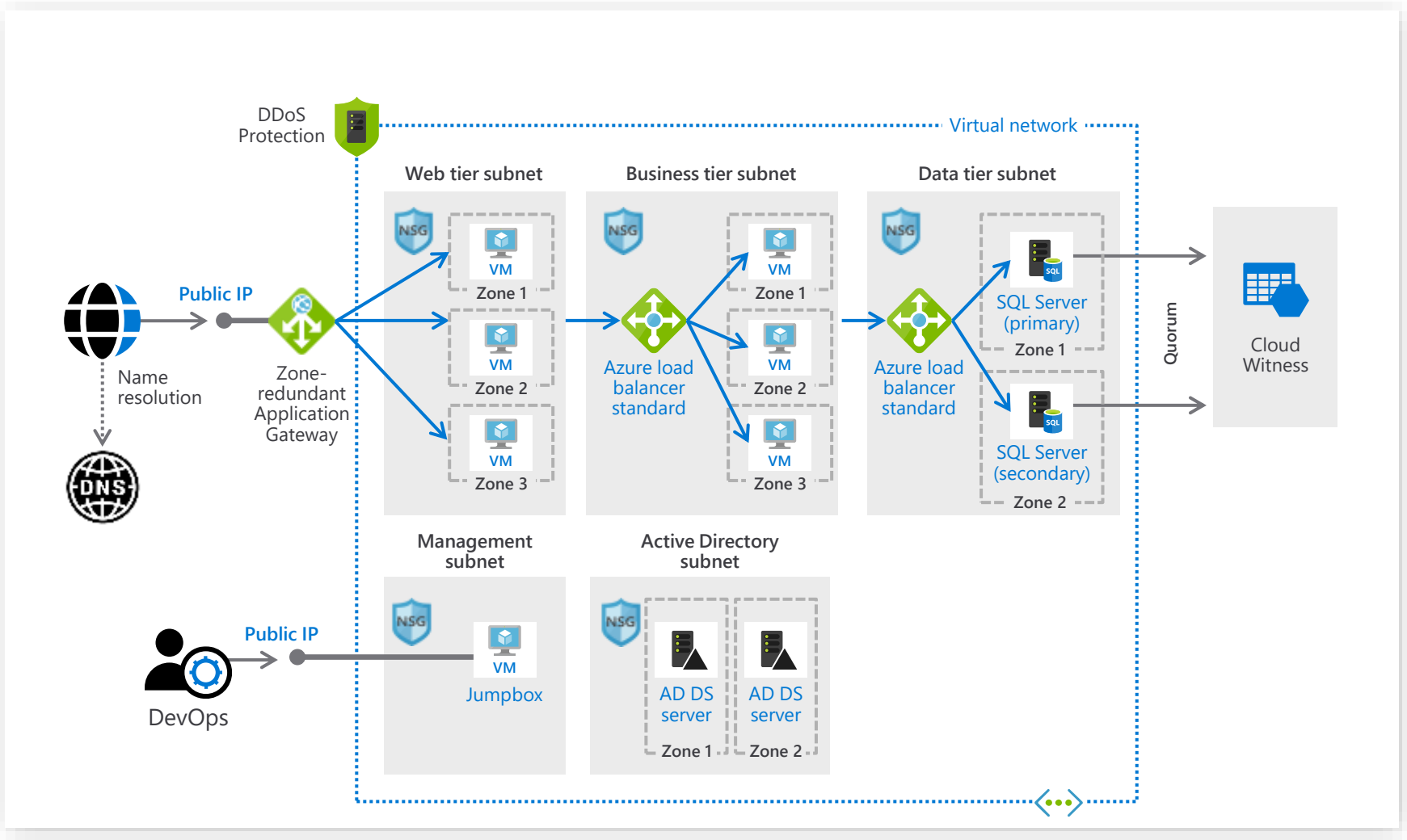


Resilience pattern: High-Availability (availability set)



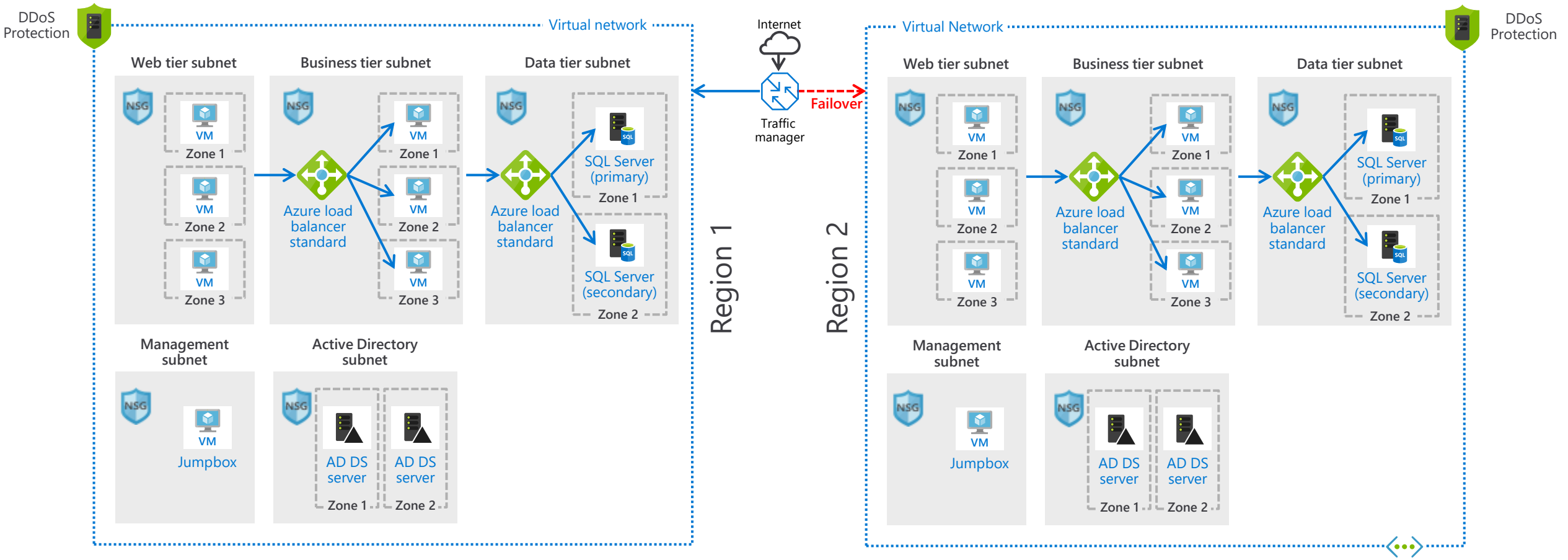
VM Availability 99.95%

Resilience pattern: High-Availability (Multiple Zones)

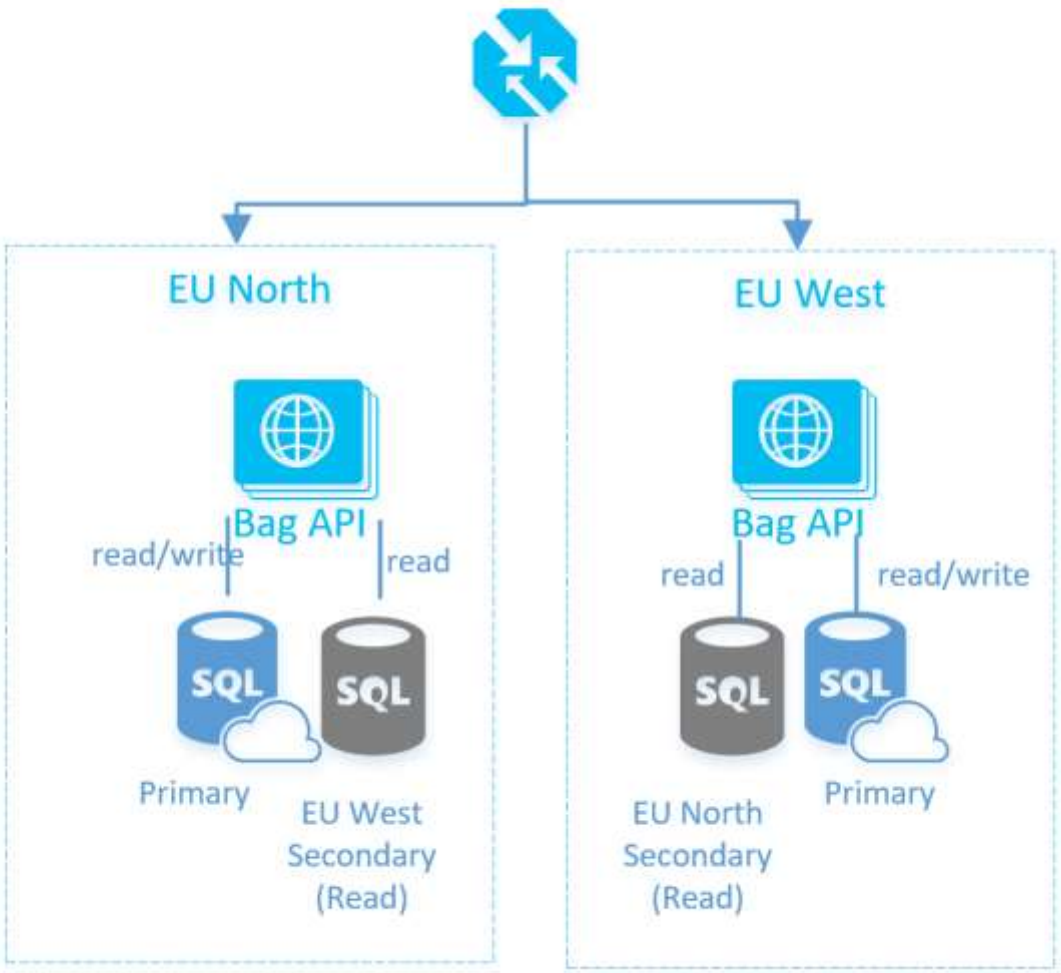


VM Availability 99.99%

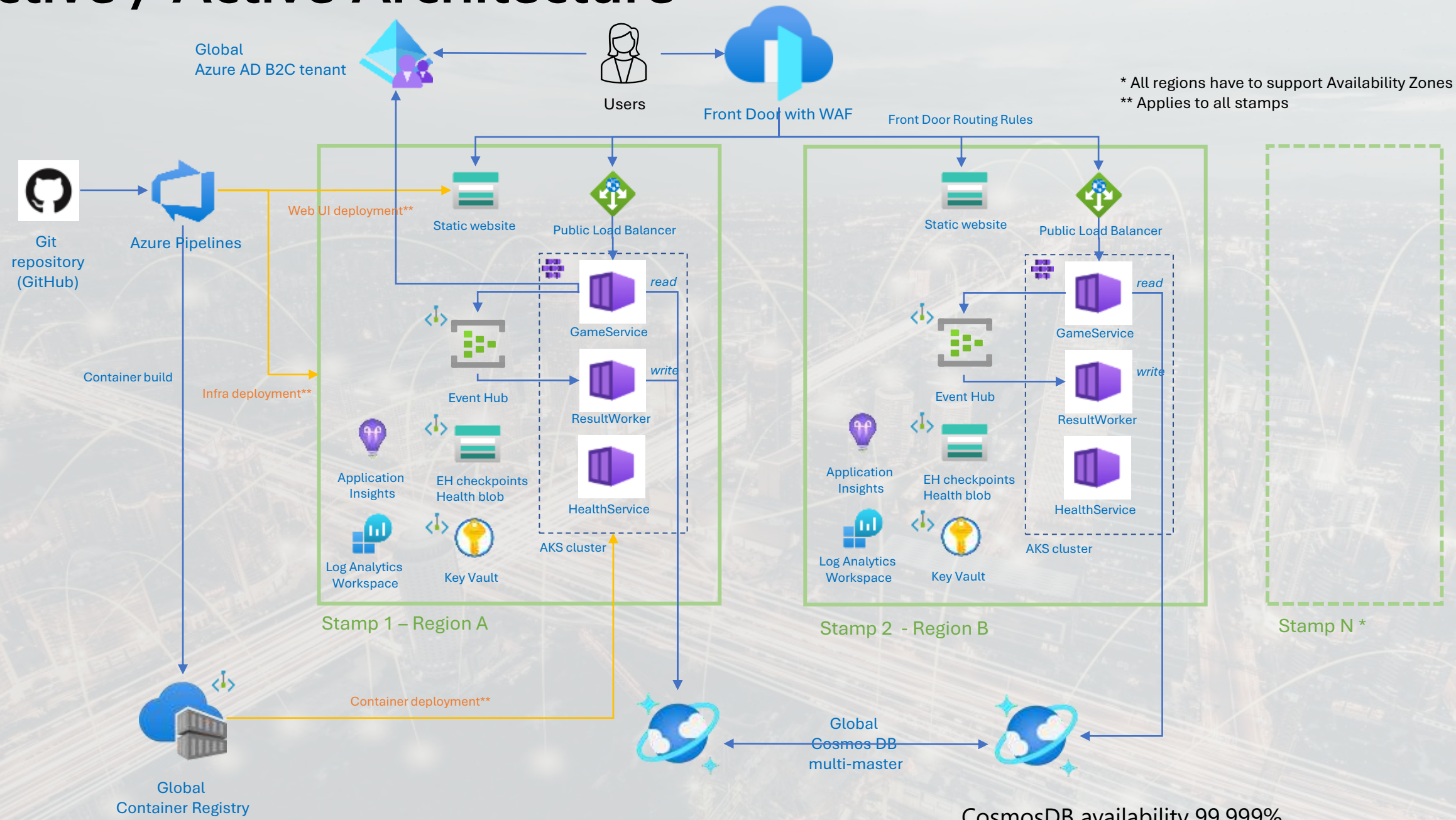
Multi Region (Active/Passive)



Architecting around a single Writable Master



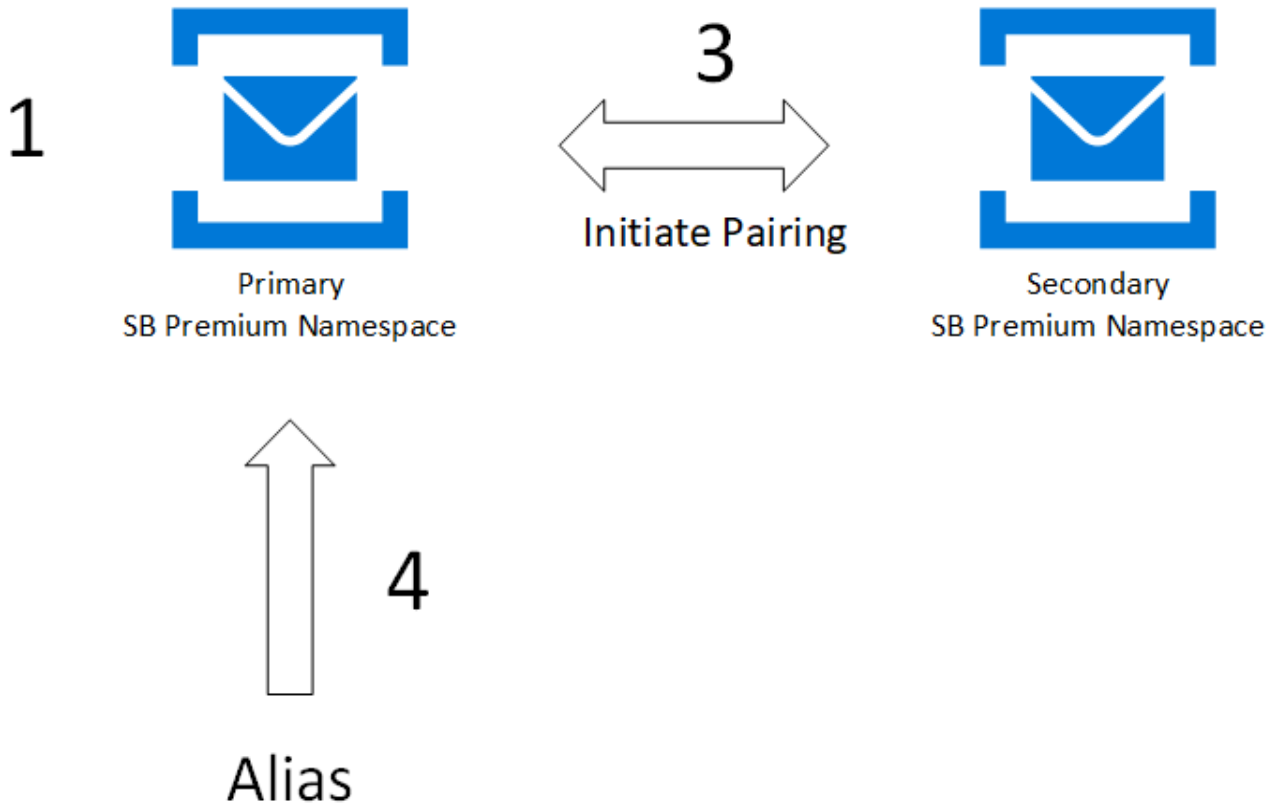
Active / Active Architecture



Multi Region Challenges

- Primary State Replication
  - Latency Constraints / Consistency ModelS
- Other State Replication – e.g. Message Stores
- Private IP Based Global Load Balancing
- Consistent Services in each region, e.g. UK South (Zone support), UK West (no Zone support)

Azure Service Bus Geo-disaster recovery



- *Alias*: The name for a disaster recovery configuration that you set up. The alias provides a single stable Fully Qualified Domain Name (FQDN) connection string. Applications use this alias connection string to connect to a namespace. Using an alias ensures that the connection string is unchanged when the failover is triggered.
- *Primary/secondary namespace*: The namespaces that correspond to the alias. The primary namespace is "active" and receives messages (this can be an existing or new namespace). The secondary namespace is "passive" and doesn't receive messages. The metadata between both is in sync, so both can seamlessly accept messages without any application code or connection string changes. To ensure that only the active namespace receives messages, you must use the alias.
- *Metadata*: Entities such as queues, topics, and subscriptions; and their properties of the service that are associated with the namespace. Only entities and their settings are replicated automatically. Messages aren't replicated.
- *Failover*: The process of activating the secondary namespace.

Feature doesn't replicate the messages held in queues or topic subscriptions or dead-letter queues

Protecting against Outages and Disasters - Service Bus Standard

- To achieve resilience against datacenter outages when using the standard messaging pricing tier, Service Bus supports two approaches: *active* and *passive* replication.
- For each approach, the given queue or topic must remain accessible in the presence of a datacenter outage. For example,
  - **contosoPrimary.servicebus.windows.net/myQueue (e.g. EU West)**
  - **contosoSecondary.servicebus.windows.net/myQueue**. (e.g. EU North)

Active Replication

- Active replication uses entities in both namespaces for every operation.
- Any client that sends a message sends two copies of the same message
  - Copy 1 sent to the primary entity (for example, **contosoPrimary.servicebus.windows.net/sales**)
  - Copy 2 sent to the secondary entity (for example, **contosoSecondary.servicebus.windows.net/sales**).
- A client receives messages from both queues. The receiver processes the first copy of a message, and the second copy is suppressed.
  - To suppress duplicate messages, the sender must tag each message with a unique identifier. Both copies of the message must be tagged with the same identifier.
  - You can use the BrokeredMessage.MessageId or BrokeredMessage.Label properties, or a custom property to tag the message. The receiver must maintain a list of messages that it has already received.
- The Geo-replication with Service Bus Standard Tier sample demonstrates active replication of messaging entities.
- **Note:** The active replication approach doubles the number of operations, therefore this approach can lead to higher cost.

Passive Replication

- In the fault-free case, passive replication uses only one of the two messaging entities.
  - A client sends the message to the active entity.
  - If the operation on the active entity fails with an error code that indicates the datacenter that hosts the active entity might be unavailable, the client sends a copy of the message to the backup entity.
  - At that point the active and the backup entities switch roles: the sending client considers the old active entity to be the new backup entity, and the old backup entity is the new active entity.
  - If both send operations fail, the roles of the two entities remain unchanged and an error is returned.
- A client receives messages from both queues. Because there is a chance that the receiver receives two copies of the same message, the receiver must suppress duplicate messages. You can suppress duplicates in the same way as described for active replication.
- In general, passive replication is more economical than active replication because in most cases only one operation is performed. Latency, throughput, and monetary cost are identical to the non-replicated scenario.

Passive Replication

- When using passive replication, in the following scenarios messages can be lost or received twice:
  - **Message delay or loss**: Assume that the sender successfully sent a message m1 to the primary queue, and then the queue becomes unavailable before the receiver receives m1. The sender sends a subsequent message m2 to the secondary queue. If the primary queue is temporarily unavailable, the receiver receives m1 after the queue becomes available again. In case of a disaster, the receiver may never receive m1.
  - **Duplicate reception**: Assume that the sender sends a message m to the primary queue. Service Bus successfully processes m but fails to send a response. After the send operation times out, the sender sends an identical copy of m to the secondary queue. If the receiver is able to receive the first copy of m before the primary queue becomes unavailable, the receiver receives both copies of m at approximately the same time. If the receiver is not able to receive the first copy of m before the primary queue becomes unavailable, the receiver initially receives only the second copy of m, but then receives a second copy of m when the primary queue becomes available.

Summary Building reliable systems is a **shared responsibility**

Customer application

Customer **app** or **workload**, built on the Azure platform.

Resiliency features

Optional Azure capabilities **a customer enables –** high availability, disaster recovery, and backup.

Resilient foundation

Core capabilities **built into the Azure platform –** how the foundation is designed, operated, and monitored to ensure availability.

Microsoft

Questions?