Microsoft

Azure SQL DB

Running a cloud database service at scale

Nigel Ellis
Distinguished Engineer
Data Platform Group (aka. SQL)

nigele@microsoft.com

July 2014

Who am I?

Distinguished Engineer in SQL team

21 years at Microsoft (all in Databases)

Worked in some form on all versions of SQL from SQL 6.5

Last 9 years mainly focused on Cloud services:

- Started in 2006 with internal focused service (CloudDB)
- My primary engineering focus is Azure SQL DB (today's talk)

I enjoy listening and learning from customers

This is my first time at SQL Bits

- Not my first time in the UK – born in Lancashire

Data Platform Continuum

Requirements for the Data Tier

Highly available database are required to support:

- Mission critical applications 7x24x365
- SaaS services with hundreds or thousands of hosted tenants

Must never lose data even in disaster situations

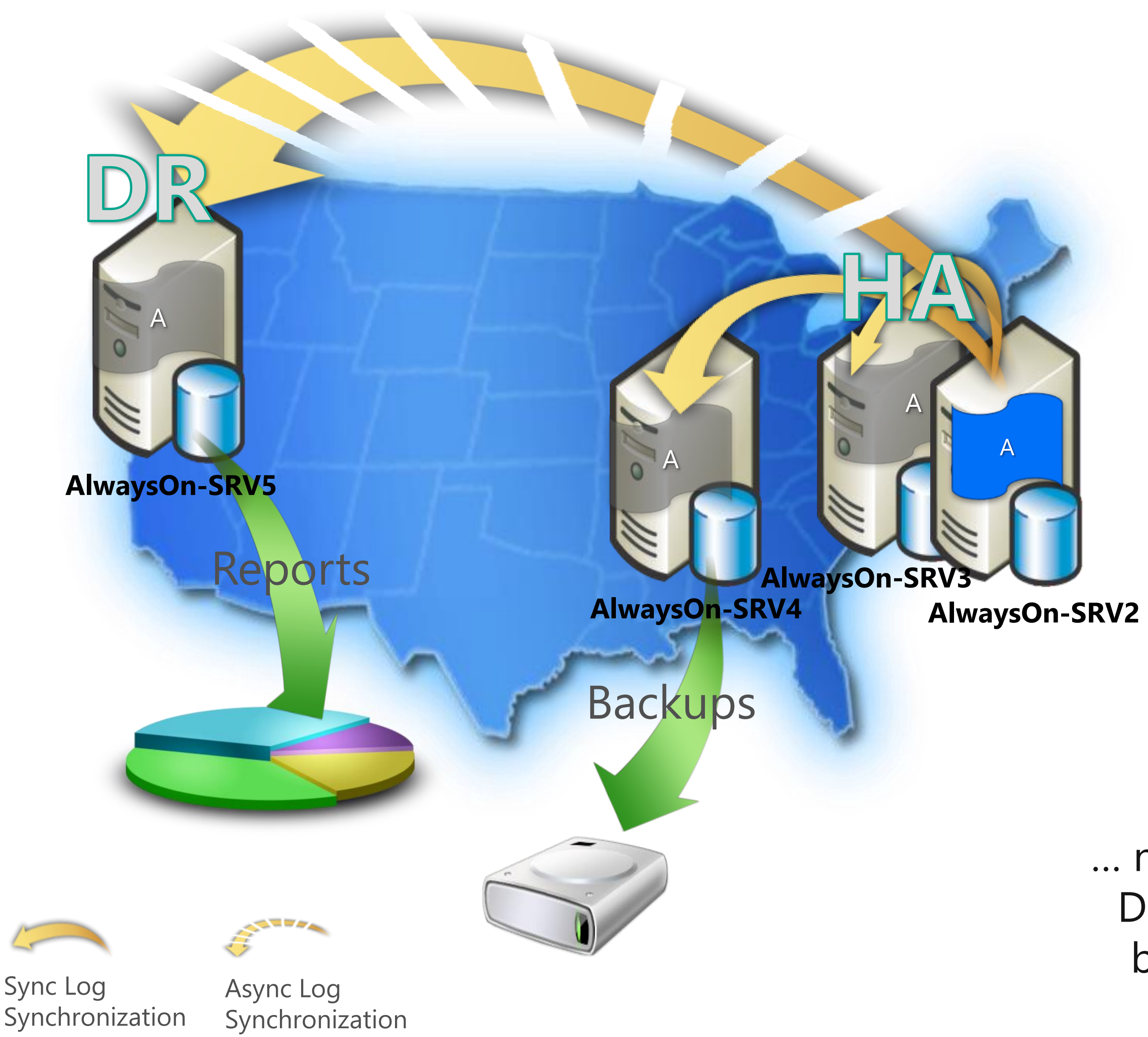
Must protect data from human errors and accidents

Must ensure fair and reasonable resource allocation

- Allocation across tenant databases must ensure predictable performance

Must be cost competitive / affordable

Build it using SQL Server



AlwaysOn for HA and GEO

Cluster needs quorum to avoid split brain

- The number of voting members determines the cluster tolerance to failures
- Can use node majority for odd # of members or majority with ties (node or file share) for even #

Cluster members must be on same Windows domain

Readable secondaries usable for read-only workloads

SQL backup/restore for redundancy

Backup scheduling

Backup storage (where?) and retention polices

Governance for Performance

EE only feature setting limits on IO, memory and CPU

Requires workload classifier (TSQL function)

... now take this **pattern** and scale to 50K... 100K, 1M DBs ...
Don't forget about tenant allocation, upgrades/patching, billing, multiple service tiers (SaaS), load-balancing, etc.

SQL Database Service Overview

A relational **database-as-a-service**, fully managed by Microsoft

For cloud-designed apps when **near-zero administration** and **enterprise-grade** capabilities are key

Perfect for cloud **architects and developers** looking for programmatic DBA-like functionality

Elastic scale & performance	Business continuity	Familiar & self-managed
Predictable performance levels	Self-service restore	Familiar tools
Programmatic scale-out	Disaster recovery	Programmatic
Dashboard views of DB metrics	Microsoft-backed SLAs	Self-managed

Where is it offered?

Azure Regions

Broad global reach

United States	US East (Virginia) US West (California) US North Central (Illinois) US South Central (Texas)
Europe	Europe North (Ireland) Europe West (Netherlands)
Asia Pacific	Asia Pacific East (Hong Kong) Asia Pacific Southeast (Singapore)

Japan	Japan East (Saitama Prefecture) Japan West (Osaka Prefecture)
Brazil	Brazil South (Sao Paulo State)
China	Beijing Shanghai

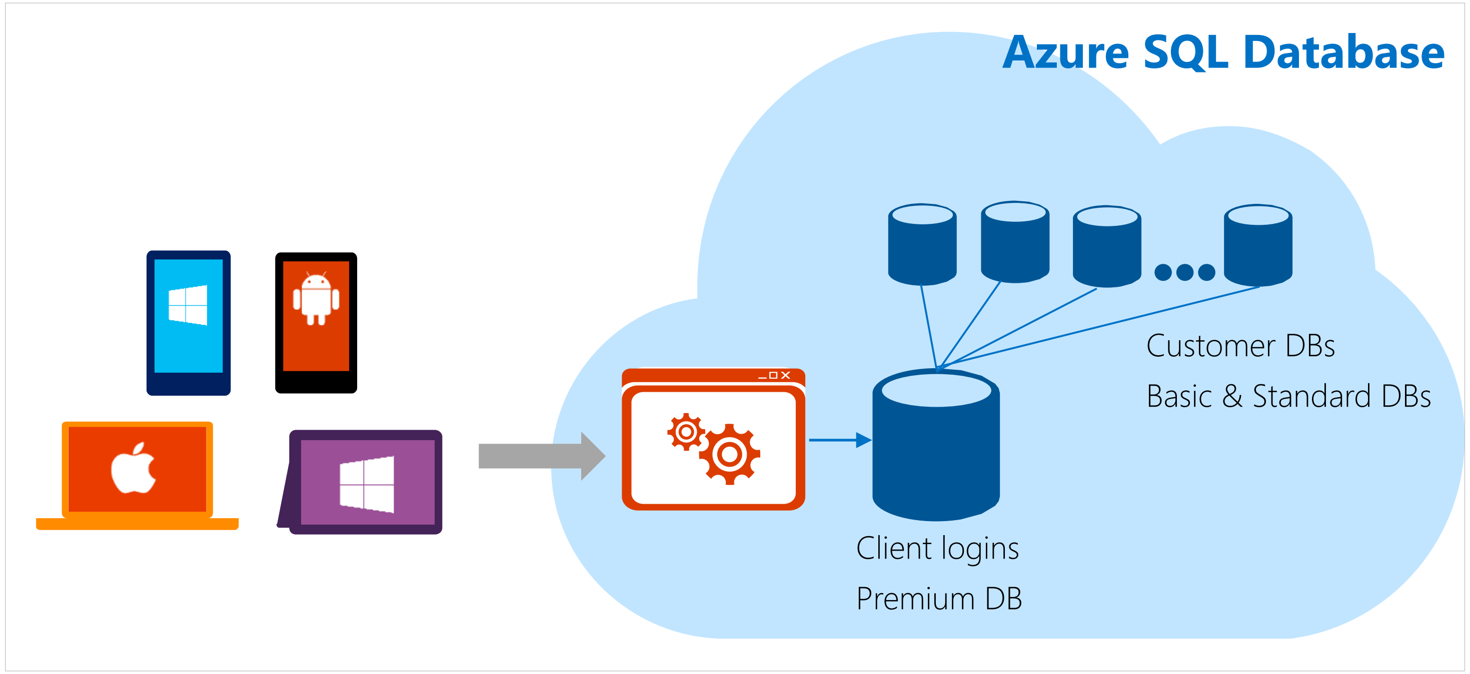
Building Software-as-a-Service Apps

Key Benefits

Customer DB isolation

Near-zero administration

Elastic scale as customers grow



"Azure gives us the ability to scale up to thousands of databases as needed… Today, more than 50 percent of new product registrations at MYOB are for our cloud accounting solutions"

Simon Raik-Allen, MYOB

MYOB

Mission-critical performance

Cost-effective scale, 10s of thousands DBs

Accelerated testing & deployment

Flavorus deployed a high volume ticketing app on Microsoft Azure and SQL Database for fast and reliable access to customers around the world

Benefits

150,000
ticket sales in 10 seconds

Ability to compete for big deals without new infrastructure investments

Improved data stability

"The way that SQL Database is architected, you just can't lose data. That's the sort of thing that makes you sleep well at night."

James Reichardt
CTO and Lead Programmer, Flavorus

Flavorus®

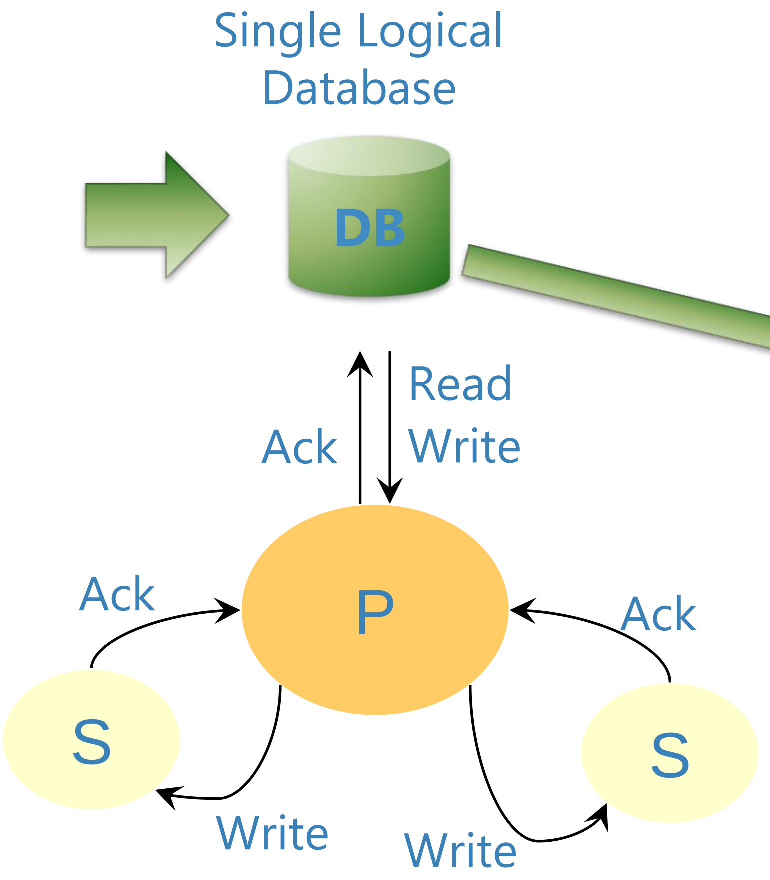
Azure SQL Database Service Tiers (in preview)

	Basic	Standard	Premium
Built for…	Light transactional workloads	Medium transactional workloads	Heavy transactional workloads
Availability SLA	99.95%*		
Database Max Size	2 GB	250 GB	500 GB
Self-Service Restore (“oops” recovery)	Any point within 7 days	Any point within 14 days	Any point within 35 days
Business Continuity	Basic recovery**	Geo-Replication, passive replica** System selected location (geo-pairing in Azure)	Active Geo-Replication, up to 4 readable replicas. Users selected location(s).
Performance Objectives	Hourly transaction rate	Transactions per minute	Transactions per second
SQL Database value prop	**App Scalability & Performance** Massive scale & performance **Business Continuity** Business continuity & data protection **Developer Efficiency** Familiar management tools & APIs, Self-managed		

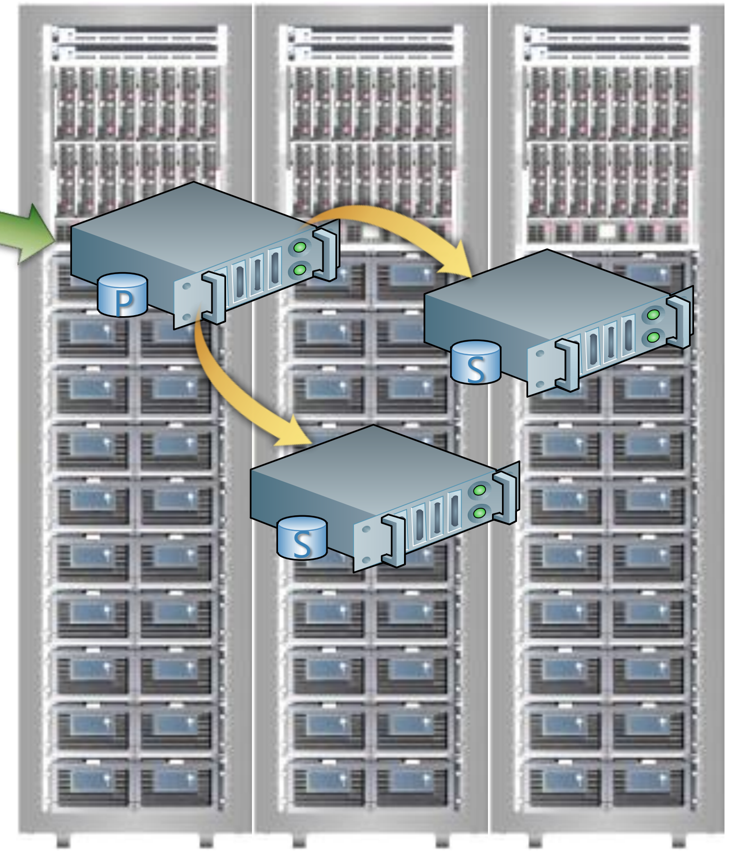
*SLAs will take effect at time of GA, Azure previews are subject to different service terms, as set forth in preview supplemental terms.

**Not all restore & disaster recovery features are available today, visit the disaster recovery documentation page to learn more.

Database High Availability



- Majority quorum up to 4 replicas
- Transparent automatic failover
- Uptime SLA of 99.95%
- Zero user or admin config



Reads are completed at the primary

Writes are replicated to secondaries

Active Geo-Replication

Mission-critical business continuity on your terms, via. programmatic APIs

Self-service activation in Premium

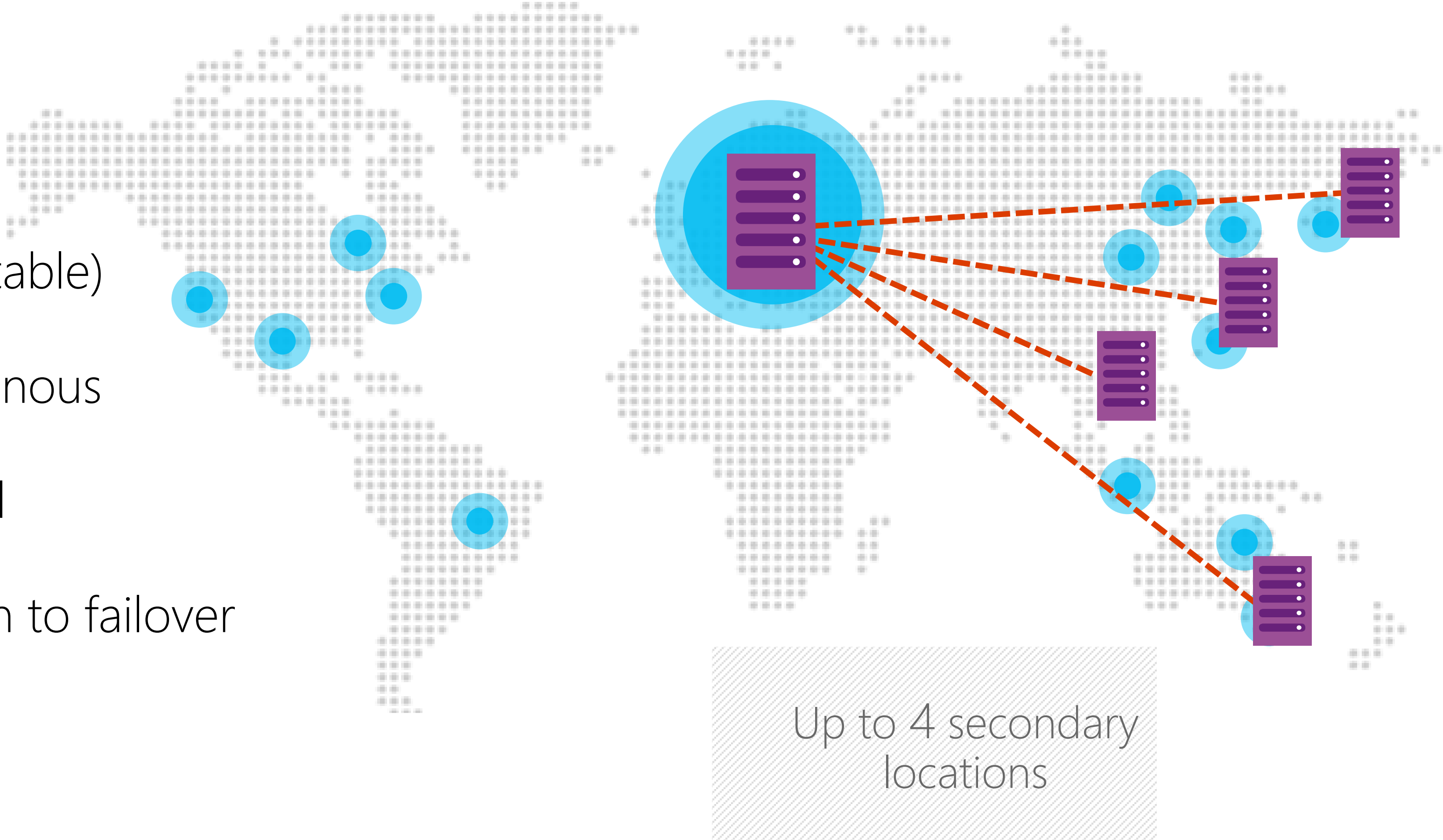
Create up to 4 readable secondaries

Replicate to any Azure region (selectable)

Automatic data replication, asynchronous

REST API, PowerShell or Azure Portal

RTO<1h, RPO<5m, you choose when to failover



Self-service restore

Programmatic “oops recovery” of data deletion or alteration

We take automatic data backups and transactional logs every 5 min (RTO)

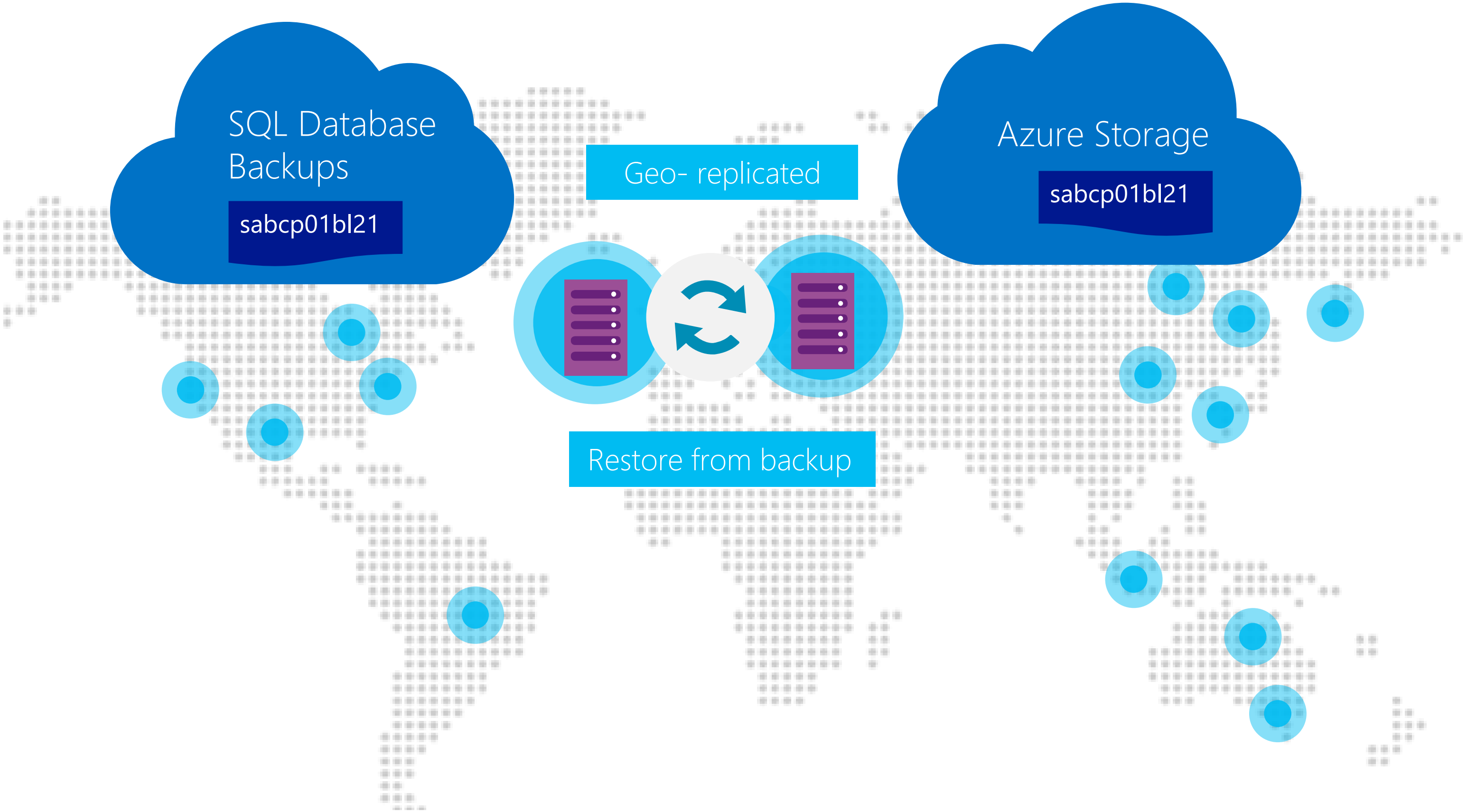
Backups pushed to Azure Storage and are geo-replicated (restore anywhere)

Recovery option creates a side-by-side database copy, non-disruptive

REST API, PowerShell or Azure Portal

Backups retention policy:

- Basic, last known state up to 24 hrs
- Standard, up to 7 days
- Premium, up to 35 days



Internals of Azure SQL DB

Engineering Requirements and Implementation

Service Topology

Service deployed by region

Each region has multiple clusters

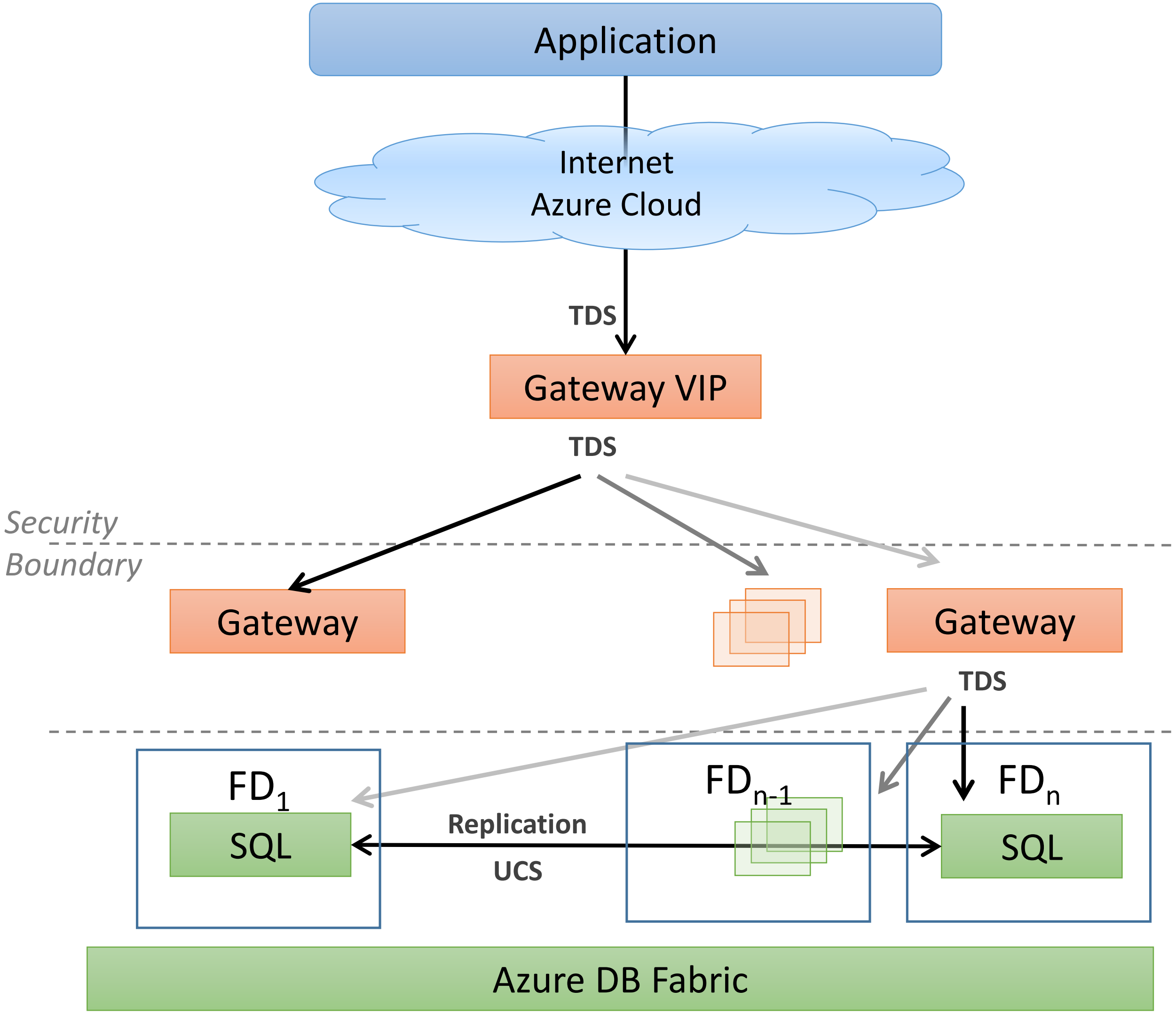
Each cluster hosts Azure Compute

Azure DB runs as Compute tenant

Typical cluster 10 – 20 racks

300 – 800 servers

13 regions worldwide, many clusters per-region



Connection & Security Model

Service exposes concept of ***logical server***

- Unit of co-location pinned to Azure ***region***
- Hosts 1 or more logical ***databases***

Clients connect directly to a database

- Large set of SQL supported within database (not instance) boundary
- Cannot hop across DBs as they are hosted on <u>different</u> backend servers

Uses regular SQL security model

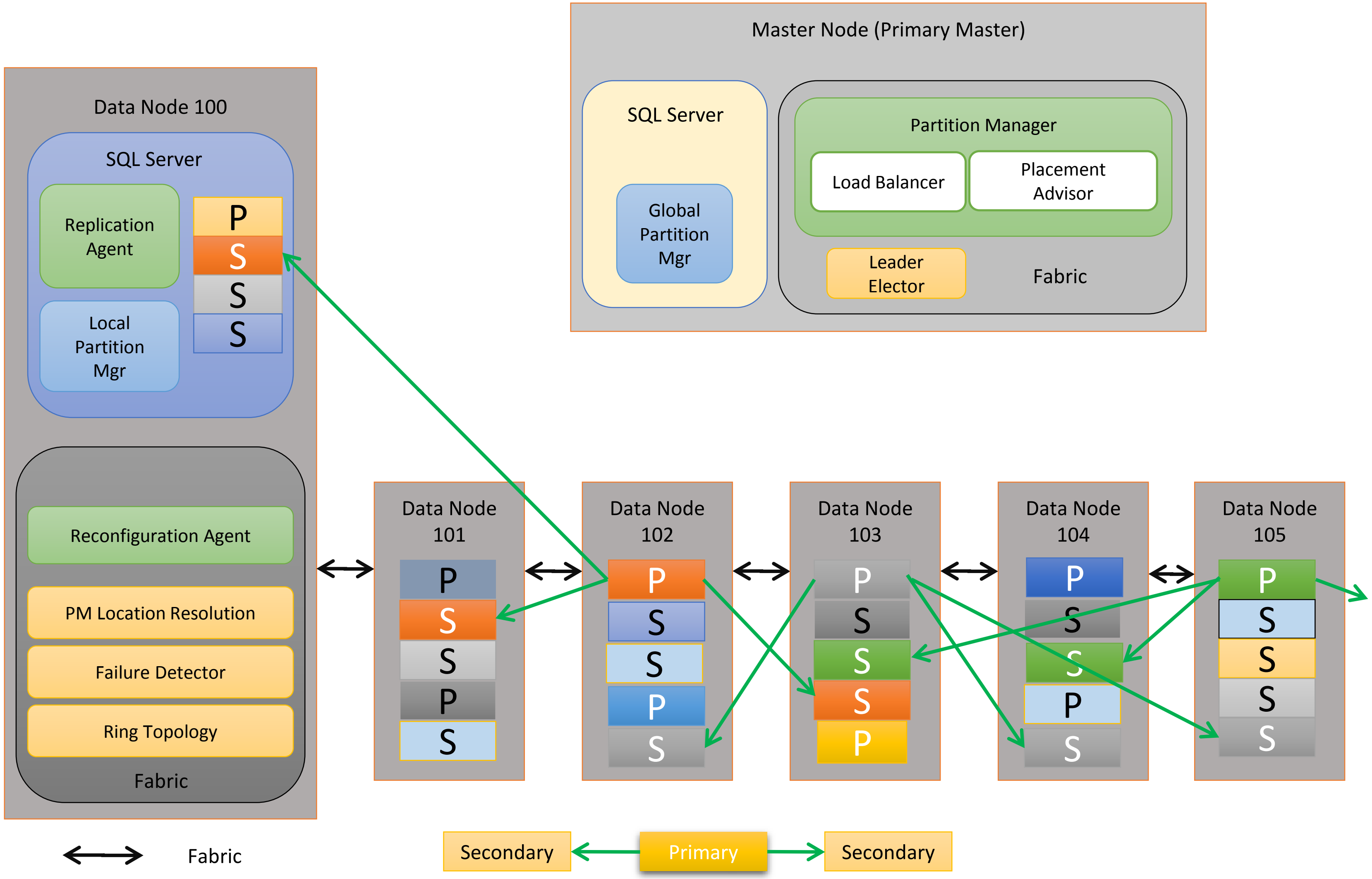
- Authenticate logins, map to users and roles
- Authorize users and roles to SQL objects

Standard SQL Auth logins

- Username + password
- Work in progress to deliver authentication with integrated security

Connection tied to target database; cannot "hop" across DBs

Components



Databases replicated with 3+ copies

Distributed across cluster of machines

Each machine hosts SQL Server and other processes

"Master" cluster controls location and provides authoritative location information in GPM

Replicas move based on failures, load changes, and cluster age

Embrace Failure: MTTR trumps MTBF

At scale the hardware failure is a routine event

- We can't blindly trust hardware and most software (including our own ☺)
- Trust but verify – example: system enforces checksums for disk & network IO
- System must protect against planned and unplanned failures

Failure modes - hard to predict gray zone failures

- Clean failure is easy to handle
- Limping along HW or a half hung process is much harder to detect
- We iterate and improve based on data (telemetry is critical)

Trade-offs between fail-fast and stay up by all means

- Not much time to wait in bad state to meet a 99.95 SLA
- Graceful vs. hard shutdown
- Always tradeoff data durability over availability, but have to meet promised RPO/RTO

Implement heuristic based repair cycle:

- Restart Process → Reboot OS → reimage OS → RMA

Dealing with Commodity Hardware

SATA drives

- On-disk cache and lack of true "write through" results in Write Ahead Logging violations
  - Force flush disk cache but causes performance degradation
- Disk failures happen daily, fail-fast on those
  - Bit-flips (Enabled page checksums to catch)
  - Drives just disappear (sometimes fixed with reboot, sometimes reseating of drives)
  - IOs are misdirected

SSD drives

- Becoming more mainstream – super fast!   Need to govern IO rate…
- Beware of wear leveling (SSDs have limited life)

Faulty NIC

- Encountered message corruption - enabled message signing and checksums on replication protocols (UCS)

Data Durability & Consistency

Data replicated within a replica set for durability and high availability

All clients need to see the same linearized order of read and write operations

Replica set is dynamically reconfigured to account for member arrivals and departures

Read-write quorums are supported and are dynamically adjusted based on replica set size

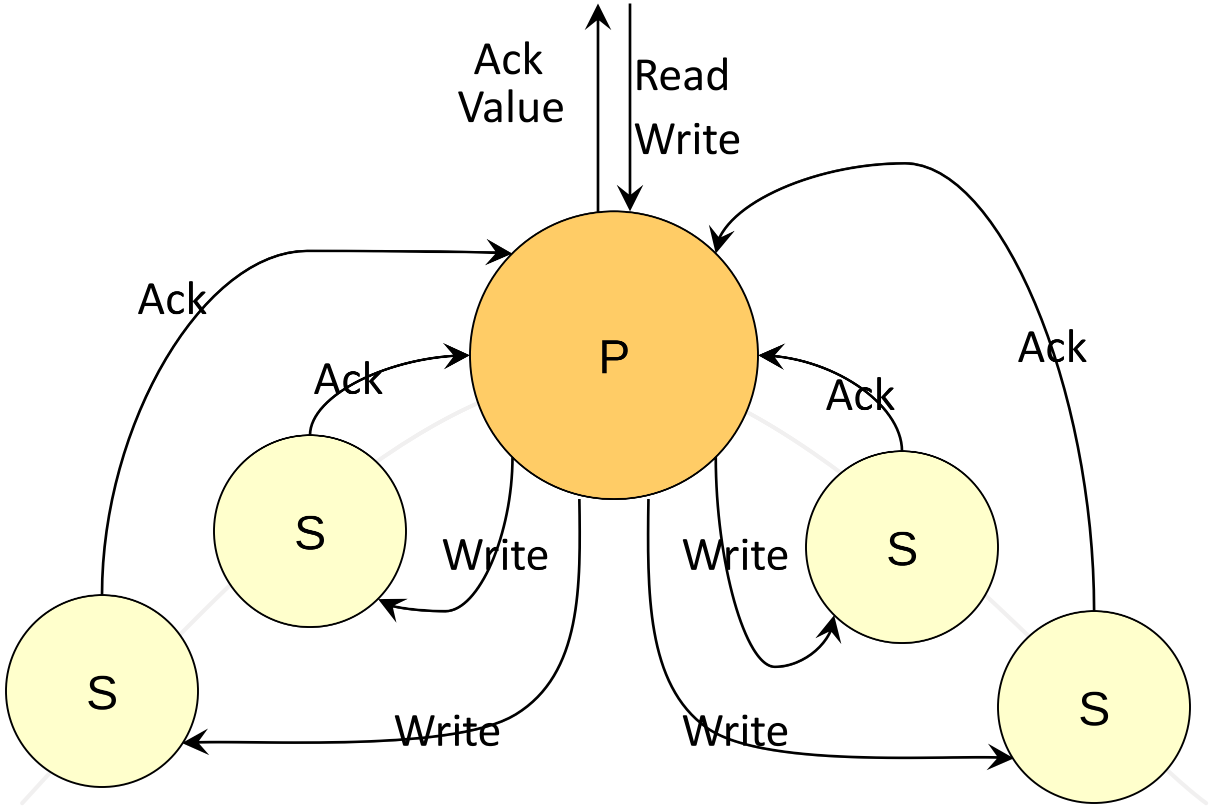
- We use a majority write quorum ($\frac{n}{2} + 1$) and a min-read quorum of 2

Replication

Reads are completed at the primary replica

Writes are replicated to the write quorum of secondaries

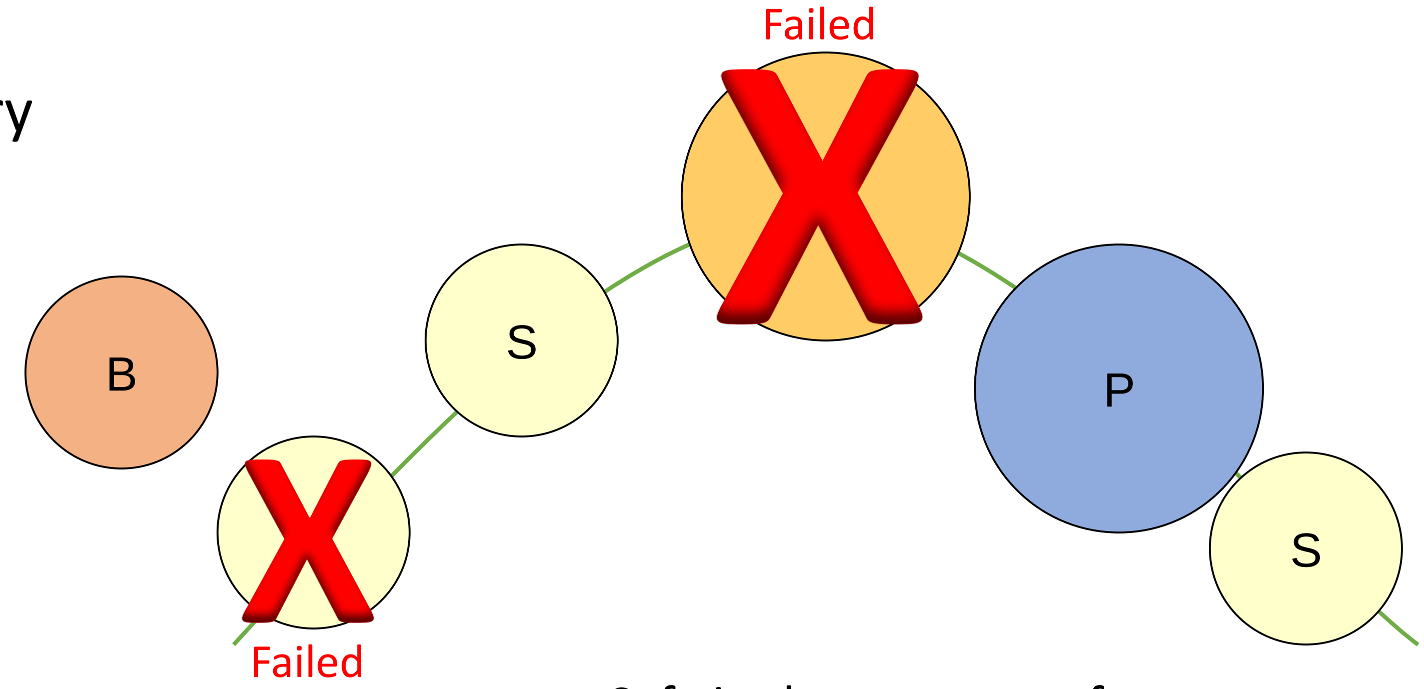
Each transaction has a commit sequence number (epoch, num)



Reconfiguration (on change)

Types of reconfiguration

- Primary failover
- Removing a failed secondary
- Adding recovered replica
- Building a new secondary



Safe in the presence of cascading failures

Assumes

- Failure detector
- Leader election

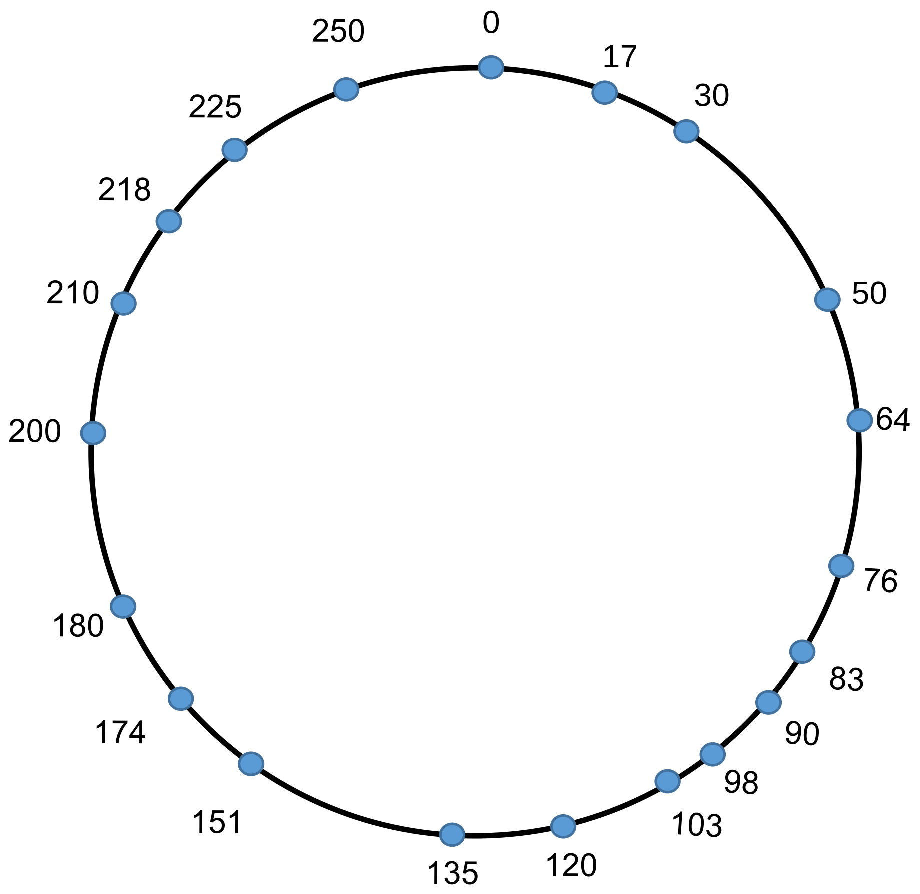
Ring Geometry

Every node is assigned a unique ID (typically a 128-bit or 160-bit number)

Active member nodes reliably form and maintain themselves in an ordered double-linked structure

The active nodes with the highest ID and lowest ID link to each other forming a ring

Rings are bootstrapped by a seed node



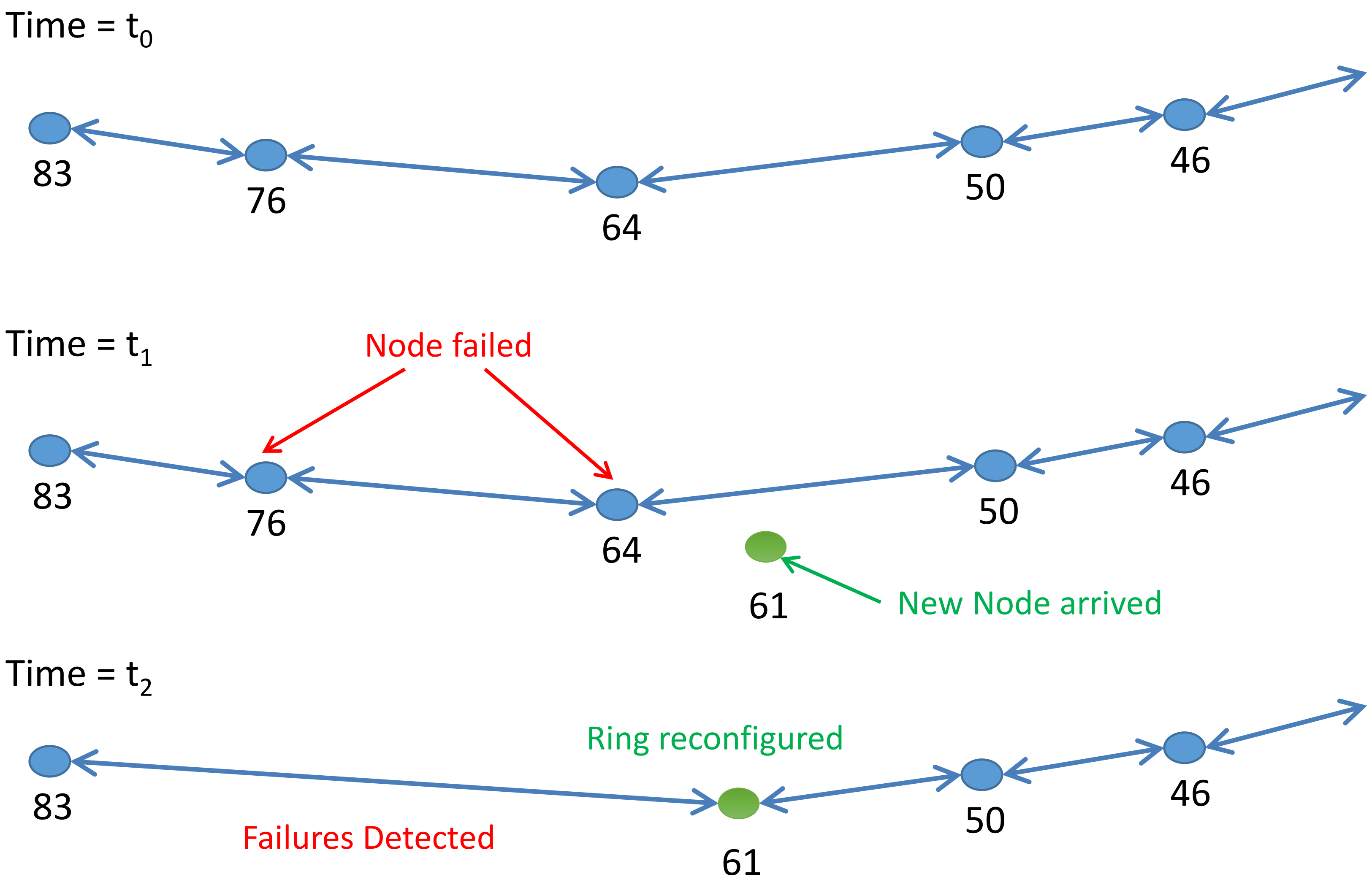
Failure Detection

Nodes establish leases and exchange 'ping' traffic to ensure liveness

Nodes can communicate directly (point-to-point) or via. other neighbors

Communication protocol forms basis of failure detection

Can detect network partitioning and other failures



Deployment and Servicing

Azure SQL DB layers over Azure Compute (built using worker roles)

OS imaged with “services” formed from SQL Server and other roles

These services built with “xcopy” installation model

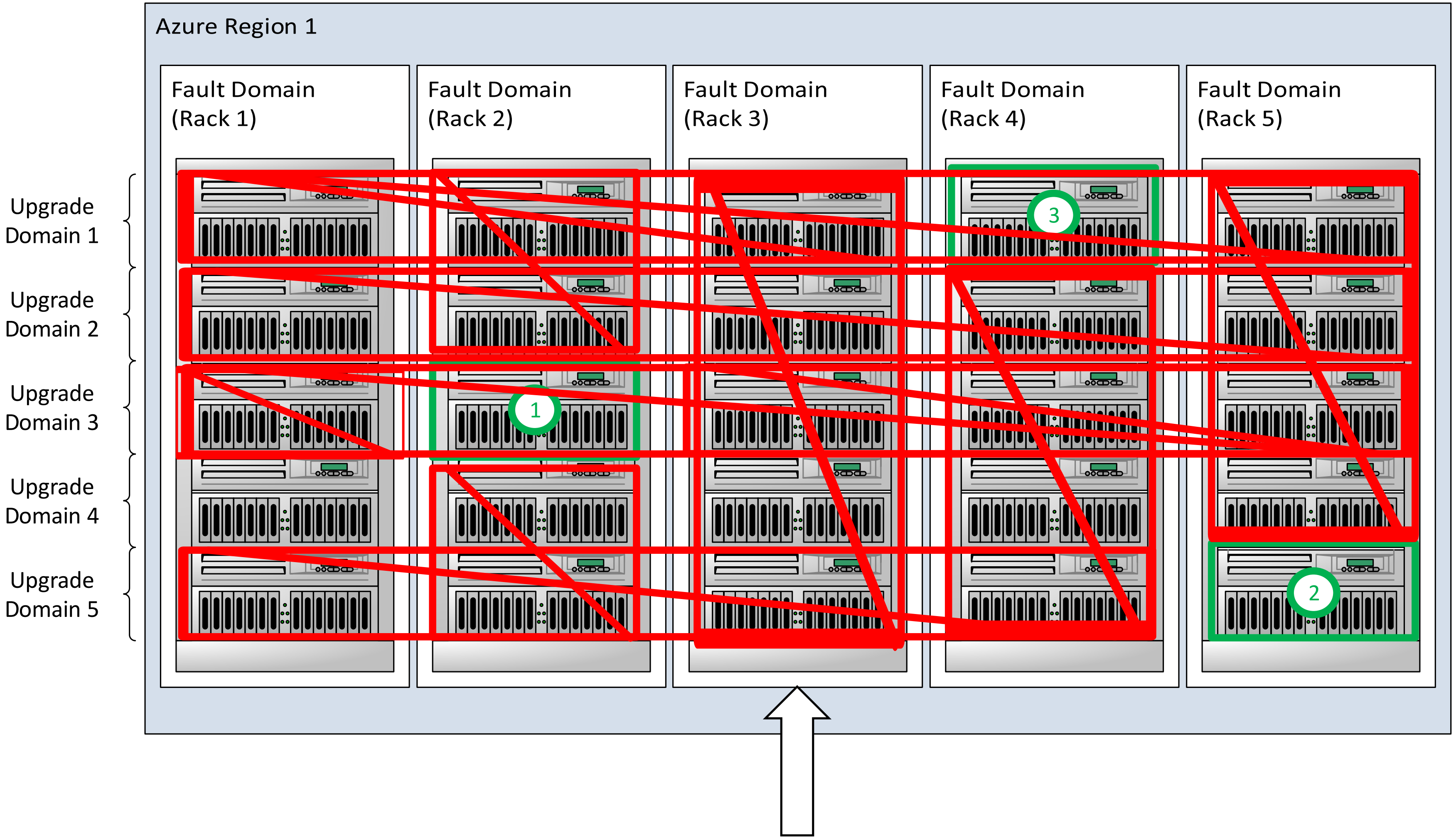
- No use of “setup” – all config read from disk
- Enables fast upgrade using side-by-side staging + switch

Upgrade is orchestrated to ensure high availability and data durability

- 4 types: hostOS, guestOS, service bits and service configuration
- Can be combined to reduce deployment time and impact

Two phase rollouts used for data format or network protocol changes

Deployment Rollout in Action



Upgrade Process / **Placement Process** … t Domain.

1. Take down the first Upgrade Domain … Fault Domain and Upgrade Domain … replica being unavailable for other replicas.
2. First UD is brought back up and waits for the machines to come back healthy. Then second UD is shut down

2. Pick the second replica from any node not already disqualified. This excludes another FD and UD.

3. Repeat until all UDs have been upgraded

3. Repeat until you have placed the required number of replicas.

Note: Upgrades can withstand a simultaneous fault domain failure with high enough spare capacity and replica count

Monitoring

Reboot/Reimage/RMA cycle for machines health/repair

All driven via. comprehensive monitoring

- Outside-in (Azure Region <> Azure Region + others)
- Inside-out (Azure Region self-monitoring)

Additional monitoring for SQL Azure services (mostly SQL engine)

- Examples: Ability to connect, Memory leaks/hung workers and Database corruption
- Trace and performance stats captured (SQL trace and DMV)
- Traces kept locally and also pushed to global region store

Monitoring drives Alerting system

- Goal is for the system to always self-heal no human intervention
- We strive for 8x5 “lights out” operation (zero drama and restful sleep)
- If healing fails, on-call team automatically paged for mitigation process

All incidents are driven via. comprehensive post-mortem system

- Focus on alerting gaps and failures in people, process and technology (see The 5 Whys)

Telemetry is king

We live and breathe data to operate the service

- At the scale we operate we cannot think about individual servers or racks
- Now getting to a point where we no longer think about single clusters
- All our actions decided based on data - a data driven culture

Telemetry on most "managed" aspects fuels our running the service

- Login availability dips raise incidents and investigations
- Databases not getting enough resources get attention
- Crashes/dumps automatically file bugs
- SQL errors give us deep insight into application and system issues

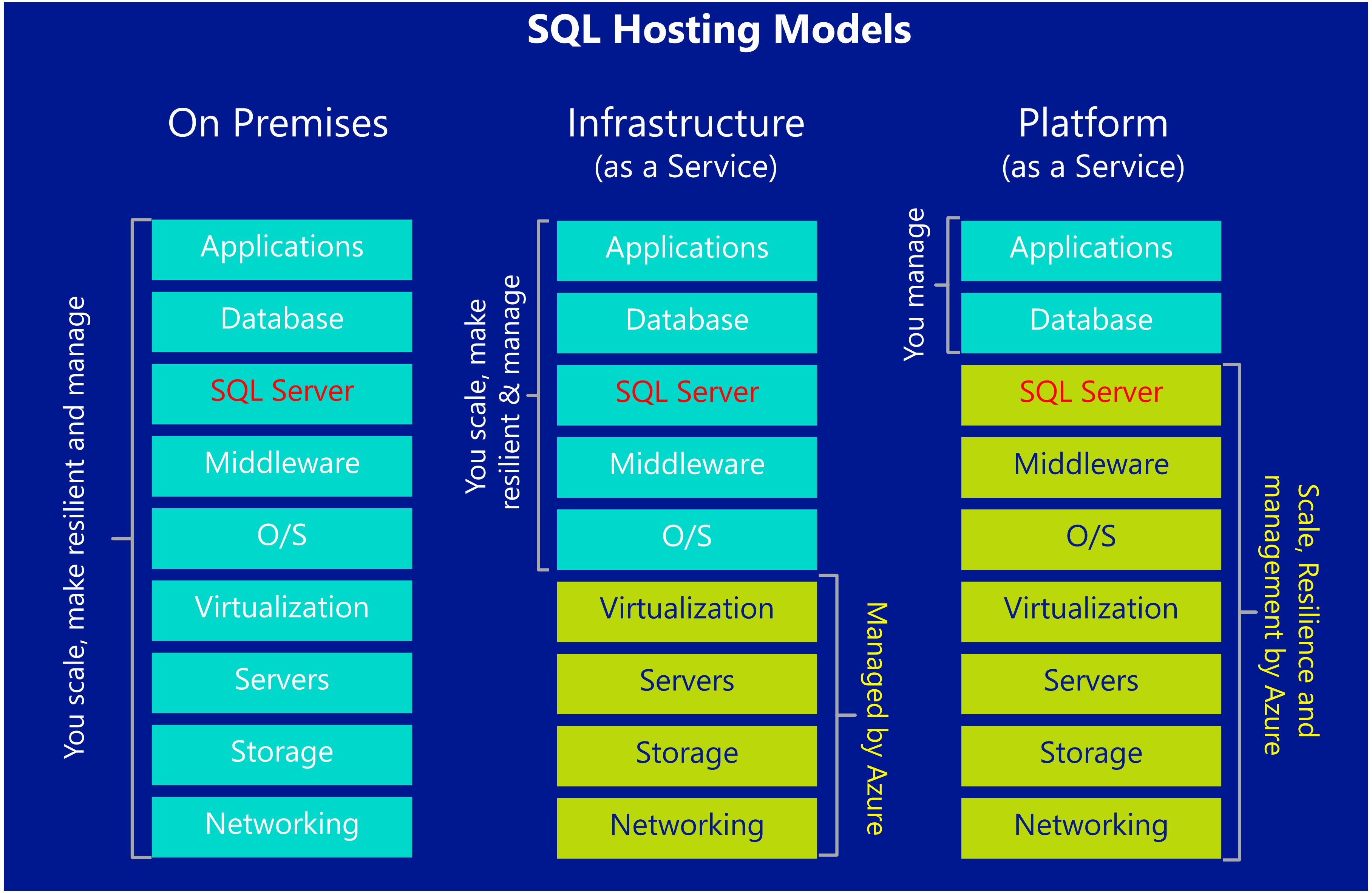
Anomaly detection & machine learning to find unexpected deviations

- Several major incidents have been averted based on anomaly detection (failure)

How we use and run our pipelines deserves a whole other talk ☺

- Make extensive use of HD Insight (HADOOP in Azure) and SQL Server
- Currently process **~200TB of telemetry per day for all Azure regions**
- Represents a HUGE learning curve – you'd think SQL Server engineers are experts at running SQL. We are getting there ☺

Hosting Choices for SQL Customers



Data Platform Continuum

Cloud 1st but not Cloud Only

Using Azure SQL DB to improve core SQL Server (features/cadence)

Many interesting (and compelling) on-premise <> Cloud scenarios

Microsoft