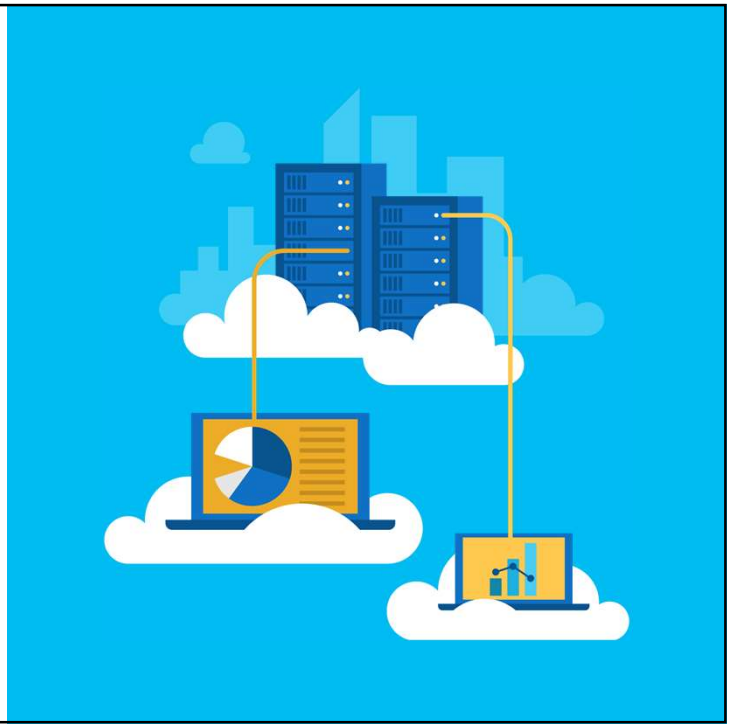
Microsoft

AZ-300T04 Module 02: Business Continuity and Resiliency in Azure

Subtitle or speaker name



1

Module 02: Business Continuity and Resiliency in Azure

Lesson 01: Business Continuity and Resiliency

2

Business Continuity and Resilience in Azure

- **Business continuity** represents the ability to perform essential business functions during and after adverse conditions such as a natural disaster or a failure of a service.
- A **technical strategy for business continuity** ensures that internal and external applications, workloads, and services are resilient by remaining operational during planned downtime and unplanned outages.
- **Architecting for resiliency in a cloud** environment focuses on failure recovery, rather than on avoiding failures. The goal is to respond to failures in a way that avoids downtime or data loss and returns applications to a fully functioning state following a failure.

3

Module 02: Business Continuity and Resiliency in Azure

Lesson 02: High Availability and Disaster Recovery

4

High Availability and Disaster Recovery

- High availability (HA) is the ability of the application to continue running in a healthy state despite localized or transient failures. Typically, high availability relies on redundancy of application components and automatic failover.
- Disaster recovery (DR) is the ability to recover from major incidents, such as service disruption that affects an entire region. Disaster recovery provisions include data backup and archiving, and may require manual intervention, such as restoring a database from backup.

5

Data Backup

- Backup is distinct from data replication.
- Data replication involves copying data in near-real-time (either synchronously or asynchronously) allowing system fail over quickly to a replica.
- Data replication can reduce the time it takes to recover from an outage by ensuring that a replica of the data is readily available.
- Data replication should not be considered as a substitute to backups.

6

Module 02: Business Continuity and Resiliency in Azure

Lesson 03: Resiliency

7

Identifying Requirements

Identify the expected recovery time objective and recovery point objective:

Recovery time objective (RTO) is the maximum acceptable time that an application can be unavailable after an incident.

- If RTO is 90 minutes, you must be able to restore the application to a running state within 90 minutes from the start of a disaster.
- If you have a very low RTO, consider a second deployment running in standby mode.

Recovery point objective (RPO) is the maximum duration of data loss that is acceptable during a disaster.

- For example, if a store data in a single database, with no replication to other databases, and perform hourly backups, you could lose up to an hour's worth of data.

8

Estimating SLA Downtime

The table below demonstrates the maximum cumulative downtime for various SLA levels:

SLA	Downtime per week	Downtime per month	Downtime per year
99%	1.68 hours	7.2 hours	3.65 days
99.9%	10.1 minutes	43.2 minutes	8.76 hours
99.95%	5 minutes	21.6 minutes	4.38 hours
99.99%	1.01 minutes	4.32 minutes	52.56 minutes
99.999%	6 seconds	25.9 seconds	5.26 minutes

Implement composite SLAs.
For example, consider an App Service web app that writes to Azure SQL Database. As of December 2018, these Azure services have the following SLAs:

- App Service Web Apps = 99.95%
- SQL Database = 99.99%

If either service fails, the whole application fails. In general, the probability of each service failing is independent, so the composite SLA for this application is 99.95% × 99.99% = 99.94%.

9

Module 02: Business Continuity and Resiliency in Azure

Lesson 04: Application Design

10

Failure Mode Analysis (FMA)

FMA is a process for building resiliency into an application early in the design stage. The goals of an FMA include:

- Identify what types of failures an application might experience.
- Capture the potential effects and impact of each type of failure on the application.
- Identify recovery strategies.

Example: For calls to an external web service / API consider the following points of failure:

Failure mode	Detection strategy
Service is unavailable	HTTP 5xx
Throttling	HTTP 429 (Too Many Requests)
Authentication	HTTP 401 (Unauthorized)
Slow response	Request times out

11

Avoiding a Single Point of Failure

Azure provides features for application redundancy… individual VMs to an entire region.

- Single VM. Azure provides an uptime SLA for individual VMs, as long as all disks of these VMs are configured to use Premium Storage. For production workloads, we recommend using two or more VMs for redundancy.
- Availability sets. To protect against localized hardware failures, such as a power unit or a network switch failing, deploy two or more VMs in an availability set. VMs in an availability set are distributed across the fault domains, so localized a hardware failure affects only one fault domain.
- Availability zones. An Availability Zone is a separate physical datacenter within an Azure region. Azure Site Recovery. Replicate Azure virtual machines to another Azure region for business continuity and disaster recovery needs.
- Paired regions. Each Azure region is paired with another region. Regional pairs are located within the same geography in order to meet data residency requirements for tax and law enforcement jurisdiction purposes.

12

Auto Scaling and Load Balancing

Cloud applications should be able to scale out by adding more instances.

Common examples of this approach include:

- Deploying two or more VMs behind a load balancer. The load balancer distributes traffic to all the VMs. If you choose Azure Application Gateway, remember that you need to provision two or more. Application Gateway instances to qualify for the availability SLA.
- Scaling out an Azure App Service app to multiple instances. App Service automatically balances load across instances.
- Using Azure Traffic Manager to distribute traffic across a set of endpoints.

13

Multi-Region Deployment

A multi-region deployment can use:

- Active-active pattern (distributing requests across multiple active instances)
- Active-passive pattern (keeping a "warm" instance in reserve, in case the primary instance fails).

Use Azure Traffic Manager to route application traffic to different regions.

- Azure Traffic Manager performs load balancing at the DNS level and will route traffic to different regions based on the traffic routing method you specify and the health of your application's endpoints.

Configure and test health probes for load balancers and traffic managers

- Traffic Manager health probe determines whether to fail over to another region.
- Load balancer determines whether to remove a VM from rotation.

14

Enhancing Security

Ensure application-level protection against distributed denial of service (DDoS) attacks.

- Azure services are protected against DDoS attacks at the network layer.
- Azure cannot protect against application-layer attacks (true user requests vs. malicious user requests).

Adhere to the principle of least privilege for access to the application's resources.

- The default for access to the application's resources should be as restrictive.
- Grant higher level permissions on an approval basis.
- Granting overly permissive access to application resources by default can result in purposely or accidentally deleted resources.

15

Additional Resiliency Tips

- **Use a message broker that implements high availability for critical transactions**
- **Design applications to gracefully degrade**
- **Throttle high-volume users**
- **Use load leveling to smooth spikes in traffic**
- **Monitor third-party services**
- **Implement resiliency patterns for remote operations where appropriate.** If your application depends on communication between remote services, follow design patterns for dealing with transient failures, such as the Retry pattern and the Circuit Breaker pattern.
  - **Retry pattern.** Transient failures can be caused by momentary loss of network connectivity, a dropped database connection, or a timeout when a service is busy.
  - **Circuit Breaker pattern.** The Circuit Breaker pattern can prevent an application from repeatedly trying an operation that is likely to fail. The circuit breaker wraps calls to a service and tracks the number of recent failures.
- **Implement asynchronous operations whenever possible**
- **Apply compensating transactions**

16

Module 02: Business Continuity and Resiliency in Azure

Lesson 05: Testing, Deployment, and Maintenance

17

Deployment and Maintenance Tasks

Automate and test deployment and maintenance tasks.

- Use Azure Resource Manager templates to automate provisioning of Azure resources.
- Use Azure Automation Desired State Configuration (DSC) to configure VMs.
- Use an automated deployment process for application code.

For App Service deployments, store configuration as app settings.

- Define the settings in your Resource Manager templates, or by using PowerShell for increased reliability

Give resources meaningful names.

- Giving resources meaningful names makes it easier to locate a specific resource and understand its role.

Organize resource groups by function and lifecycle.

- Resource groups should contain resources that share the same lifecycle to manage deployments, delete test deployments, and assign access rights.
- Create separate resource groups for production, development, and test environments.
- In a multi-region deployment allocate resources for each region into separate resource groups.

18

Infrastructure as Code and Immutable Infrastructure

Infrastructure as code is the practice of using code to provision and configure infrastructure.

- Infrastructure as code may use a declarative approach or an imperative approach (or a combination of both).
- Resource Manager templates constitute an example of a declarative approach.
- PowerShell scripts constitute an example of an imperative approach.

Immutable infrastructure is the principle that you shouldn't modify infrastructure after it's deployed to production.

19

Maximize Application Availability

Design release processes to maximize application availability. Use the blue/green or canary release deployment techniques to deploy applications to production.

- Blue-green deployment is a technique where an update is deployed into a production environment separate from the live application. After you validate the deployment, switch the traffic routing to the updated version.
- Canary releases are similar to blue-green deployments. Instead of switching all traffic to the updated version, you roll out the update to a small percentage of users, by routing a portion of the traffic to the new deployment. If there is a problem, back off and revert to the old deployment.

20

Module 02: Business Continuity and Resiliency in Azure

Lesson 06: Data Management

21

Replicating Data

Geo-replicate databases.

- Azure SQL Database and Azure Cosmos DB both support geo-replication, which enables you to configure secondary database replicas in other regions.
- With Azure SQL Database, you can create auto-failover groups, which facilitate automatic failover.

Geo-replicate data in Azure Storage.

- Data in Azure Storage is automatically replicated within a datacenter.
- For higher availability, use Read-access geo-redundant storage (RA-GRS), which replicates your data to a secondary region and provides read-only access to the data in that region.

For VMs, do not rely on RA-GRS replication to restore the VM disks (VHD files).

- Use Azure Backup and consider using managed disks.
- Managed disks provide enhanced resiliency for VMs in an availability set, because the disks are sufficiently isolated from each other to avoid single points of failure.

22

Additional Data Management Considerations

Below are further considerations for managing data:

- **Sharding.** Consider using sharding to partition a database horizontally.
- **Optimistic concurrency and eventual consistency.** Transactions that block access to resources through locking (pessimistic concurrency) can cause poor performance and considerably reduce availability.
- **Document data source fail over and fail back processes, and then test it.** In the case where your data source fails catastrophically, a human operator will have to follow a set of documented instructions to fail over to a new data source.
- **Periodic backup and point-in-time restore.** Regularly and automatically back up data and verify you can reliably restore both the data and the application
- **Ensure that no single user account has access to both production and backup data.** Data backups are compromised if one single user account has permission to write to both production and backup sources.
- **Validate your data backups.** Regularly verify that your backup data is what you expect by running a script to validate data integrity, schema, and queries.

23

Module 02: Business Continuity and Resiliency in Azure

Lesson 07: Monitoring and Disaster Recovery

24

Best Practices for Monitoring and Alerting Applications

Below is a list best practices for monitoring and alerting applications:

1. Implement best practices for monitoring and alerting in your application.
2. Measure remote call statistics and make the information available to the application team.
3. Track the number of transient exceptions and retries over an appropriate timeframe.
4. Track the progress of long-running workflows and retry on failure.
5. Implement application logging.
   - Log in production.
   - Log events at service boundaries. Include a correlation ID that flows across service boundaries. If a transaction flows through multiple services and one of them fails, the correlation ID will help you pinpoint why the transaction failed.
   - Use semantic logging, also known as structured logging. Unstructured logs make it hard to automate the consumption and analysis of the log data, which is needed at cloud scale.
   - Use asynchronous logging. With synchronous logging, the logging system might cause the application to fail, as incoming requests are blocked while waiting for log writes.
6. Implement logging using an asynchronous pattern.

25

Test the Monitoring Systems

Plan for and test disaster recovery

- Create an accepted, fully-tested plan for recovery from any type of failure that may affect system availability.
- Choose a multi-site disaster recovery architecture for any mission-critical applications.
- Identify a specific owner of the disaster recovery plan, including automation and testing.
- Ensure the plan is well-documented, and automate the process as much as possible.
- Establish a backup strategy for all reference and transactional data, and test the restoration of these backups regularly.
- Train operations staff to execute the plan, and perform regular disaster simulations to validate and improve the plan.

Implement operational readiness testing

- If the application fails over to a secondary region then perform an operational readiness test before fail back to the primary region.

Perform data consistency checks

- If a failure happens in a data store, there may be data inconsistencies when the store becomes available again, especially if the data was replicated.

26

27