A. Traffic Manager

Azure Traffic Manager is a DNS-based global traffic distribution service that directs client requests to the most appropriate endpoint based on policies like performance, priority, or geographic location. It works at the **DNS level**, which means it doesn’t actively proxy traffic like Front Door or Application Gateway, but instead returns the correct IP address for the client to connect to. It's commonly used for high-availability setups and global distribution scenarios.

Traffic Manager supports different **routing methods**, including **Priority**, **Weighted**, **Performance**, and **Geographic**. These methods allow you to control traffic flow across regions based on response time or location. It uses **health probes** with configurable intervals and timeouts to monitor endpoint health. If a probe fails consistently, Traffic Manager stops directing traffic to that endpoint and reroutes to a healthy one.

To implement it, you define **profiles** that include endpoint types (Azure services, external endpoints, nested profiles), set the routing method, and configure probe parameters. While Traffic Manager is ideal for DR and load distribution, it operates on the DNS layer, so changes are subject to DNS caching. This means failover isn’t truly instant, but it's still much faster and more efficient than manual DNS updates.

B. Front Door

Azure Front Door is a global, scalable **application delivery network** that acts as a reverse proxy and load balancer for web apps. Unlike Traffic Manager, which operates at the DNS level, Front Door works at **Layer 7 (HTTP/HTTPS)** and directly proxies traffic to the backend, providing faster failover and richer routing capabilities. It is ideal for high-traffic, performance-sensitive apps that need fast global delivery and secure routing.

Front Door shares similarities with **Application Gateway** (like supporting SSL termination and WAF), but while App Gateway is regional and used primarily within a VNet, Front Door is a **global entry point**. It can route traffic to multiple regions based on latency, priority, or even custom rules, and it provides automatic failover with near-zero downtime during a regional disaster.

To implement it, you define **Front End domains** (custom or Azure-provided), create **Backend Pools** (with regional endpoints or App Services), and configure **Routing Rules** to determine how requests are handled. It also supports health probes to automatically remove failing endpoints and integrate WAF policies. Front Door is a key component in modern global DR strategies due to its performance, routing flexibility, and integrated security.

C. Azure Cost Management

Azure Cost Management is a suite of tools provided by Microsoft to help organizations **track, analyze, and optimize** their cloud spending. It provides visibility into consumption-based costs, allowing users to understand where their budget is going and take action to reduce waste. It works across subscriptions and tenants, including support for multi-cloud environments like AWS via Cost Management + Billing.

You can review your **invoices** and drill down into charges using **Cost Analysis**, which breaks down spending by resource, service, or tag. With this, you can identify unexpectedly high costs, set up chargeback models for departments, and monitor cost trends over time. The **Budgets** feature lets you set spending limits and receive alerts when approaching or exceeding those thresholds, helping you proactively manage cloud expenses.

Azure also integrates **Advisor Recommendations** directly into Cost Management. These include suggestions to resize or deallocate underused resources, switch to reserved instances, or eliminate unused services. Together, these tools support cost transparency, governance, and optimization—essential practices for any team scaling its use of Azure or managing cloud infrastructure efficiently.

1. What is DR and Which Apps in Azure Require It?

Disaster Recovery (DR) is the set of strategies and technologies used to ensure business continuity in the event of a catastrophic failure, such as a regional outage, hardware failure, or human error. In Azure, DR ensures that data and application functionality can be restored with minimal downtime and data loss, depending on the organization's Recovery Point Objective (RPO) and Recovery Time Objective (RTO). The goal is to minimize service disruption and safeguard critical business operations.

Not all applications require the same level of DR. Mission-critical apps—such as financial systems, transactional apps, and systems handling sensitive or essential services—require robust DR setups with minimal RPO and RTO values. Non-critical systems, like internal tools or staging environments, may tolerate higher RPO/RTO or even manual recovery procedures. Azure provides native support for DR in many services, but planning which apps need it and to what extent is a crucial architectural decision.

---

2. Hot/Cold DR

Hot and Cold DR refer to different levels of readiness and cost for disaster recovery environments. A **Hot DR** setup means that the secondary region is always running, synchronized with the primary, and ready to take over immediately with little or no downtime. This approach ensures low RTO and RPO but incurs significantly higher costs due to duplicate resources running 24/7.

On the other hand, a **Cold DR** setup keeps resources in the secondary region turned off or even unprovisioned until needed. Data might be replicated, but compute resources are activated only during a failover. This method is more cost-effective but results in higher RTO since time is needed to bring everything online. Organizations often choose a **Warm DR** compromise, where some services are kept active and others are staged for activation, balancing cost and recovery speed.

---

3. RPO (Recovery Point Objective) / RTO (Recovery Time Objective)

RPO defines the maximum acceptable amount of data loss measured in time. For instance, an RPO of 15 minutes means your system should tolerate losing no more than 15 minutes of data. Services like Azure SQL and Cosmos DB offer very low (even zero) RPO through features like geo-replication, while other services, such as MySQL, may only support RPOs of 5 minutes or more with custom configurations.

RTO represents the maximum acceptable amount of downtime following a disaster. If your RTO is 30 minutes, your disaster recovery plan must ensure services are up and running again within that timeframe. Organizations must align RPO/RTO targets with business needs, service tiers, and budgets—lowering either one typically increases infrastructure costs. Azure supports achieving tight RPO/RTO through paired regions, replication strategies, and high-availability architectures.

---

4. How Do We Implement DR?

Implementing disaster recovery in Azure typically involves three main steps: restoring data, activating compute resources in a secondary region, and modifying routing so users are redirected to the new location. The implementation depends on the service type and the criticality of the workload. For services with native geo-redundancy (like Azure SQL or Cosmos DB), data can be quickly accessed in a secondary region.

If compute resources like VMs or App Services are not pre-provisioned in the secondary region, they must be deployed and configured as part of the recovery process. This takes time and adds to the RTO, so for apps requiring quick recovery, compute resources should be kept in a warm or hot state. Lastly, **routing changes** must be made—either manually (e.g., updating DNS records) or automatically (via Traffic Manager or Front Door)—to ensure users reach the active instance.

---

5. DR of Data

For disaster recovery of data, Azure provides different solutions with varying RPOs. Services like **Azure SQL, Cosmos DB, and Azure Storage** support **RPO = 0 minutes**, thanks to built-in geo-replication and failover support. This means no data is lost in a disaster scenario, and these services can quickly be brought online in a paired region. Geo-redundancy is generally enabled by default or easily configurable.

Other services, like **Azure MySQL**, do not support instant failover in the same way. For these, **RPO ≥ 5 minutes** is typical. This means you must recreate the database in the secondary region and **restore data from a backup or replicated storage account**. This adds time to recovery, making it more suitable for less-critical workloads or those that can tolerate some data loss and delay in restoration.

---

6. DR of Compute

Compute resources (VMs, App Services, Kubernetes, etc.) must be considered separately from data. In DR terms, compute relates directly to **RTO**—how fast services are back online. If **RTO = 0 minutes**, it implies active-active or hot standby compute infrastructure, which is always running and ready for failover. This comes at a higher cost but guarantees minimal downtime.

In contrast, if your application can tolerate some downtime (**RTO > 0 minutes**), you may choose to have **inactive or no compute resources** in the secondary region, spinning them up only during a failover. This reduces cost but increases recovery time. Azure tools like Azure Site Recovery can automate the failover of VMs, while IaC tools like ARM, Bicep, or Terraform can help quickly deploy necessary resources.

---

7. Routing in DR

Routing is a critical aspect of DR—it determines how quickly users are redirected to a functioning backup region. One option is to **manually inform users** of a new address or URL. While simple, this approach is slow, error-prone, and unsuitable for public-facing or mission-critical apps. A better option is to **manually update DNS records** to point to the IP of the resources in the secondary region.

However, the most robust option is **automatic routing** using Azure services like **Traffic Manager** and **Front Door**. These services continuously probe endpoints for availability and can reroute traffic based on defined rules when failure is detected. Automatic routing significantly reduces RTO and can seamlessly direct users to the active region without any manual intervention, making it ideal for high-availability scenarios.
