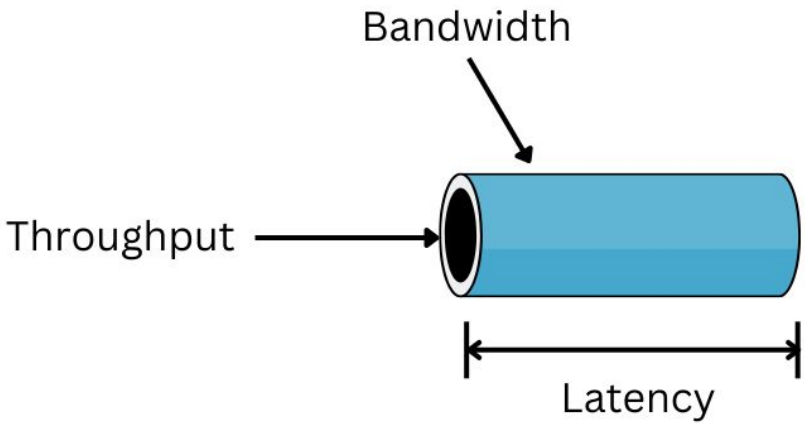
Domain 3: Design High-Performing Architectures

What is high-performing?

- **High performing**: The ability of your application to efficiently handle and process tasks, under heavy workloads.
- **Throughput**: The amount of work the system can accomplish in a given amount of time
- **Bandwidth**: The maximum amount of data that can be transferred over a network connection in a specific period of time.
- **Latency**: The amount of time it takes for a job to completed in a given amount of time.



High Performing Storage (S3 vs. EFS vs. EBS)

		File Amazon EFS	**Object** Amazon S3	**Block** Amazon EBS
Performance	Per-operation latency	Low, consistent	Low, for mixed request types, and integration with CloudFront	Lowest, consistent
	Throughput scale	Multiple GBs per second	Multiple GBs per second	Single GB per second
Characteristics	Data Availability/Durability	Stored redundantly across multiple AZs	Stored redundantly across multiple AZs	Stored redundantly in a single AZ
	Access	One to thousands of EC2 instances or on-premises servers, from multiple AZs, concurrently	One to millions of connections over the web	Single EC2 instance in a single AZ
	Use Cases	Web serving and content management, enterprise applications, media and entertainment, home directories, database backups, developer tools, container storage, big data analytics	Web serving and content management, media and entertainment, backups, big data analytics, data lake	Boot volumes, transactional and NoSQL databases, data warehousing & ETL

High Performing Compute

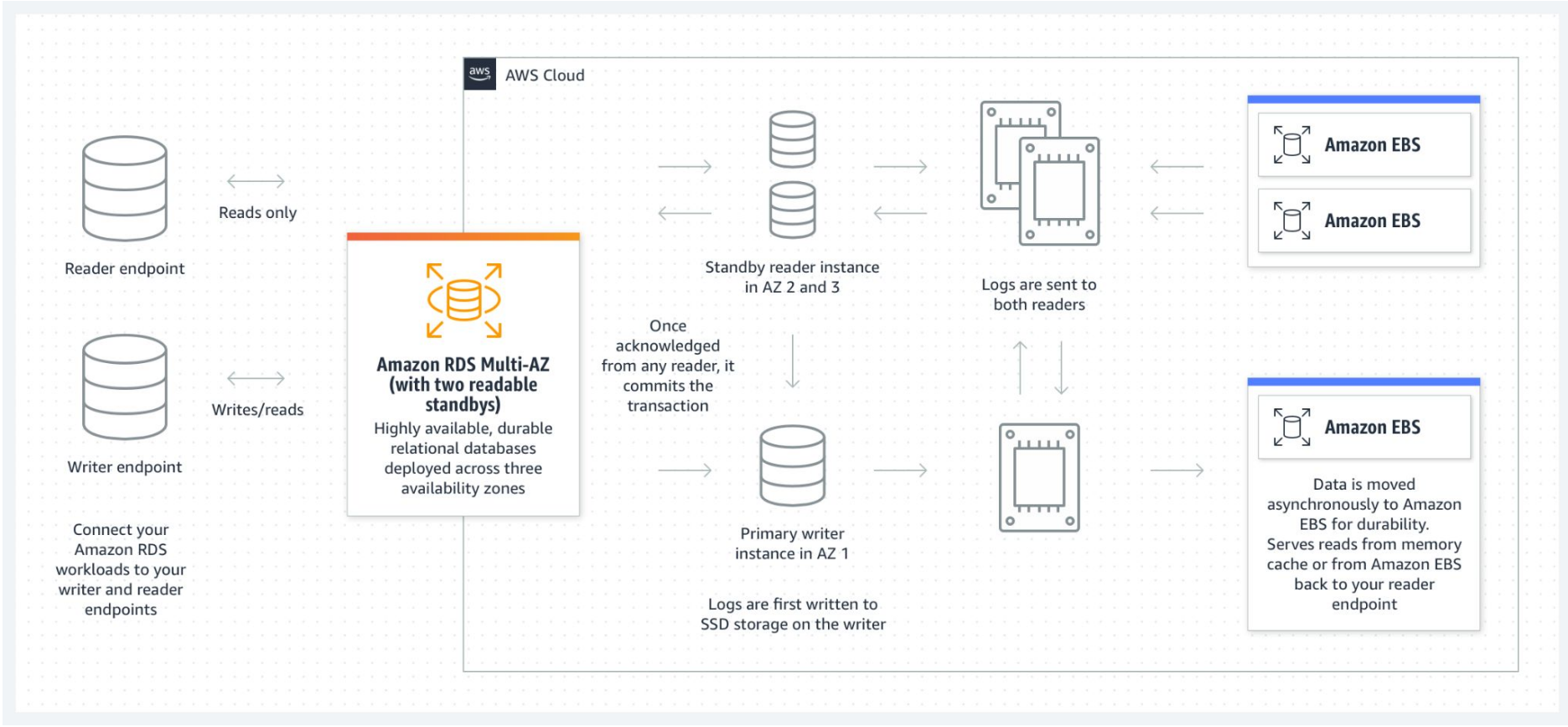
- **AWS Batch:** Has the ability to run batch jobs (i.e. multiple jobs at once). It is useful for running multiple ML/data analytics jobs at once.
- **Edge services:** Edge services bring computing power closer to you. For example, AWS Outpost provides the user a physical AWS computer to run their applications.
- **Publish / Subscribe**: By using queues (Simple Queue Service), multiple servers can process jobs from the queues that they are “subscribed” to.
- **Auto-Scaling:** Allows the number of running servers to increase/decrease based on load, CPU usage, etc.
- **Serverless:** Allows for more flexible applications that can easily scale based on demand
    - **Lambda**: Runs code when an event is triggered. The underlying servers are invisible to the application.
    - **ECS Fargate**: Fargate is a feature of ECS (Elastic Container Service) that allows containers to be run in a serverless fashion.

Database Types

- **Relational:** store data in structured tables with rows and columns, and they use a schema to define the structure of the data. They are based on the relational model and use SQL (Structured Query Language) for querying and manipulating data.
    - Amazon RDS can run most relational database engines (MySQL, PostgreSQL, etc.)
    - Amazon has their own database engine called Aurora which provides performance improvements

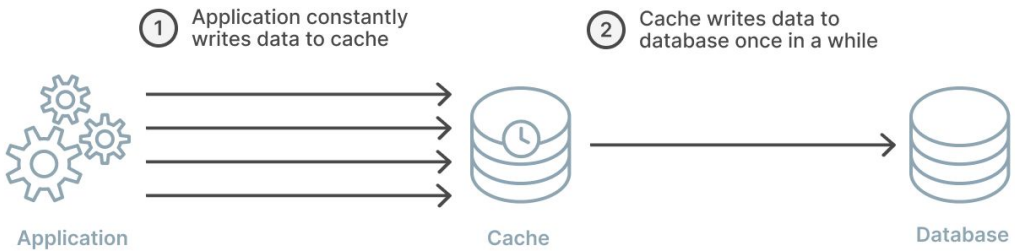
- **Non-Relational (NoSQL):** Databases that are designed to handle large volumes of unstructured or semi-structured data and offer flexible schemas that can adapt to changing data requirements. Many different subtypes:
    - Document databases: MongoDB, Amazon DocumentDB
    - Key-value stores: Amazon DynamoDB, Redis
        - The preferred NoSQL solution
    - Graph Databases: Amazon Neptune
    - In-memory Databases: Amazon Elasticache

Database Features



Database Caching

- **Benefits**:
  - **Improved Performance**: By serving data directly from the cache, database caching reduces the need to fetch data from disk or perform complex query processing, resulting in faster response times and improved application performance.
  - **Reduced Latency**: Caching reduces the latency associated with accessing data from the database, especially for frequently accessed data, leading to a better user experience.
  - **Scalability**: Caching helps offload the database by reducing the number of queries and load on the database server, allowing it to handle a larger volume of requests without performance degradation.



Category: CSAA – Design High-Performing Architectures

A Solutions Architect needs to deploy a mobile application that collects votes for a singing competition. Millions of users from around the world will submit votes using their mobile phones. These votes must be collected and stored in a highly scalable and highly available database which will be queried for real-time ranking. The database is expected to undergo frequent schema changes throughout the voting period.

Which of the following combination of services should the architect use to meet this requirement?

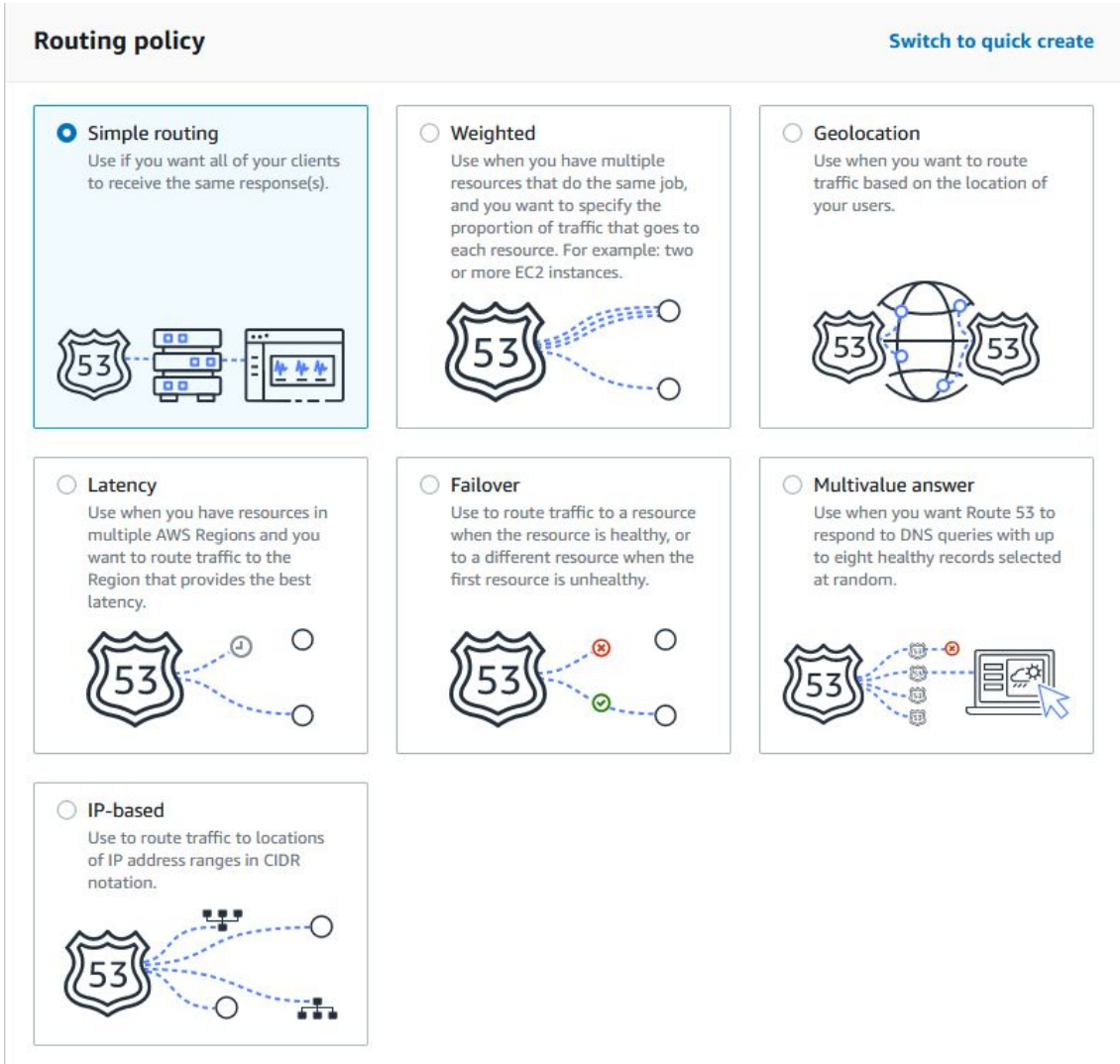
- Amazon Aurora and Amazon Cognito
- Amazon Relational Database Service (RDS) and Amazon MQ
- Amazon DynamoDB and AWS AppSync
- Amazon DocumentDB (with MongoDB compatibility) and Amazon AppFlow

- Amazon Aurora and Amazon Cognito
- Amazon Relational Database Service (RDS) and Amazon MQ
- Amazon DynamoDB and AWS AppSync
- Amazon DocumentDB (with MongoDB compatibility) and Amazon AppFlow

Route 53

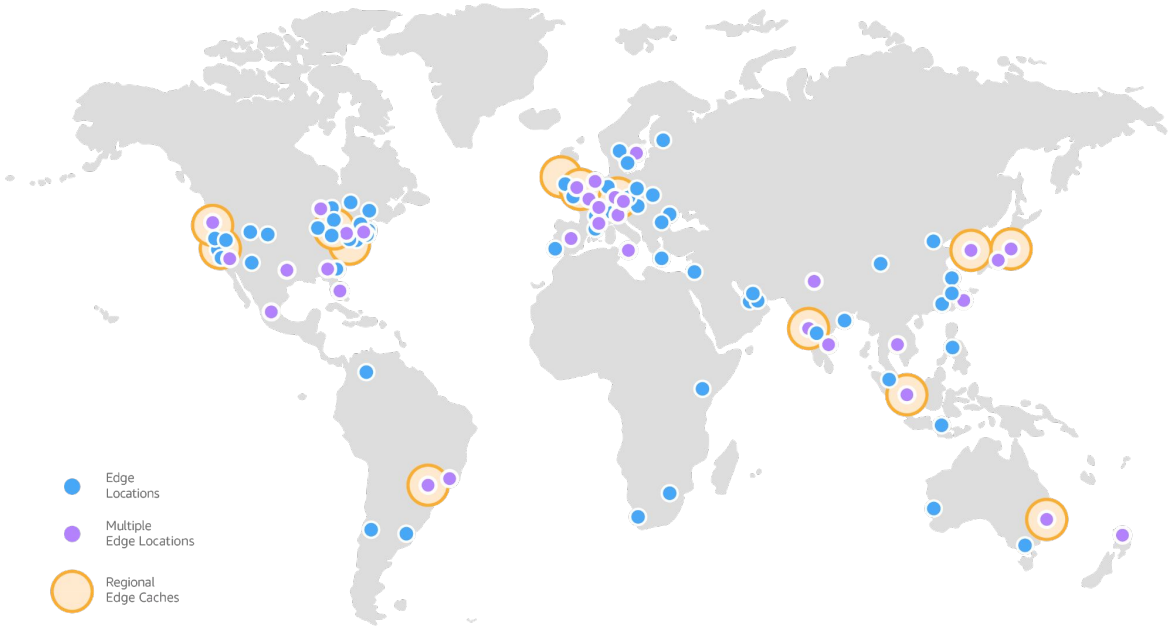
- Domain Name System (DNS) Service
- Allows you to route traffic based on some rules (routing policies)
- Different from a load balancer
- “Load balancers can monitor targets that span multiple availability zones but not multiple regions. Route 53 monitors your targets regardless of their location, as long as they are reachable by Route 53.”

ELB Health Checks vs Route 53 Health Checks For Target ...Tutorials Dojohttps://tutorialsdojo.com › AWS

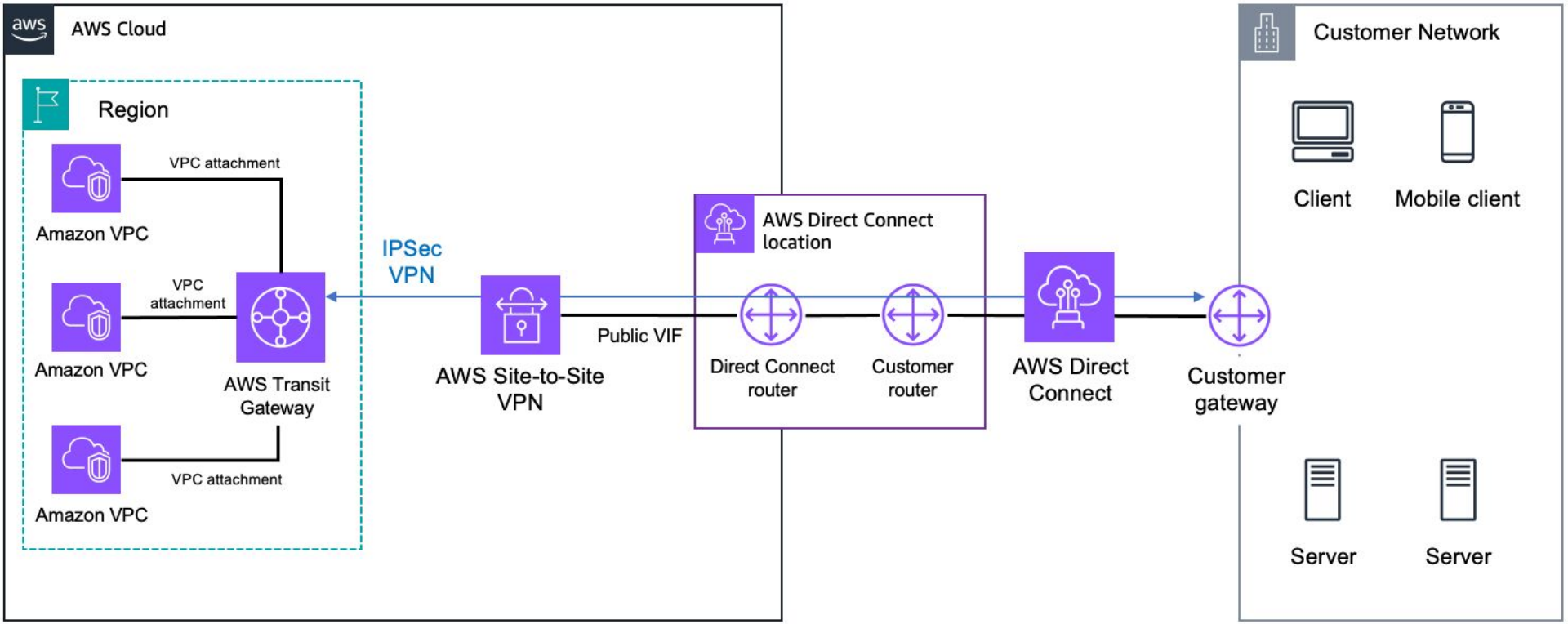


Cloudfront

- Your content is cached and distributed across a global network of edge locations
- When a user requests content from your website, CloudFront automatically routes the request to the nearest edge location, reducing latency and improving the user experience.

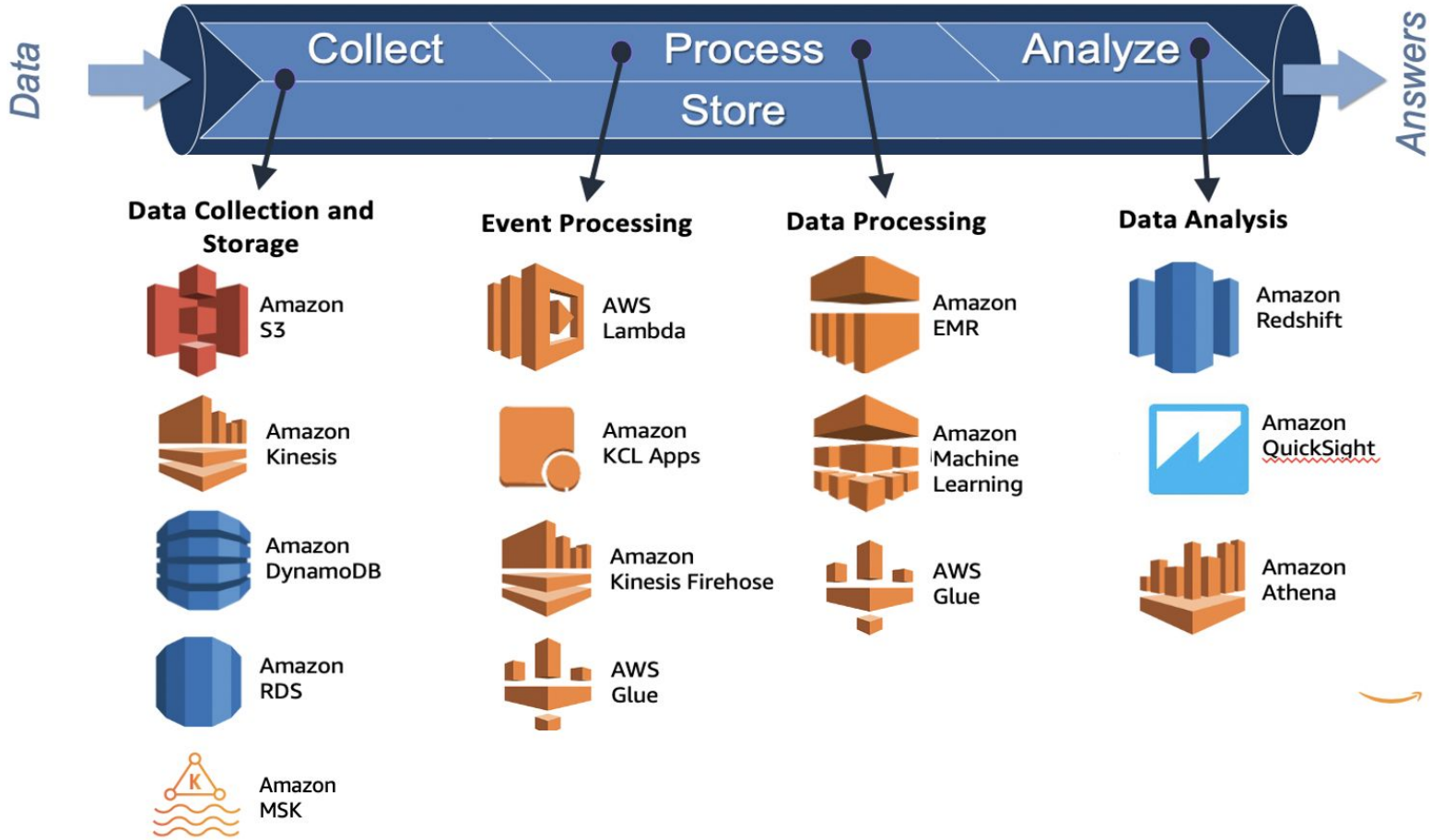


AWS VPN / Direct Connect



- [x] Use Route 53 Geolocation Routing policy.
- [ ] Set up an Application Load Balancers that will automatically route the traffic to the proper AWS region.
- [ ] Use Route 53 Weighted Routing policy.
- [ ] Set up a new CloudFront web distribution with the geo-restriction feature enabled.

Handling Data



Hands-on Demo