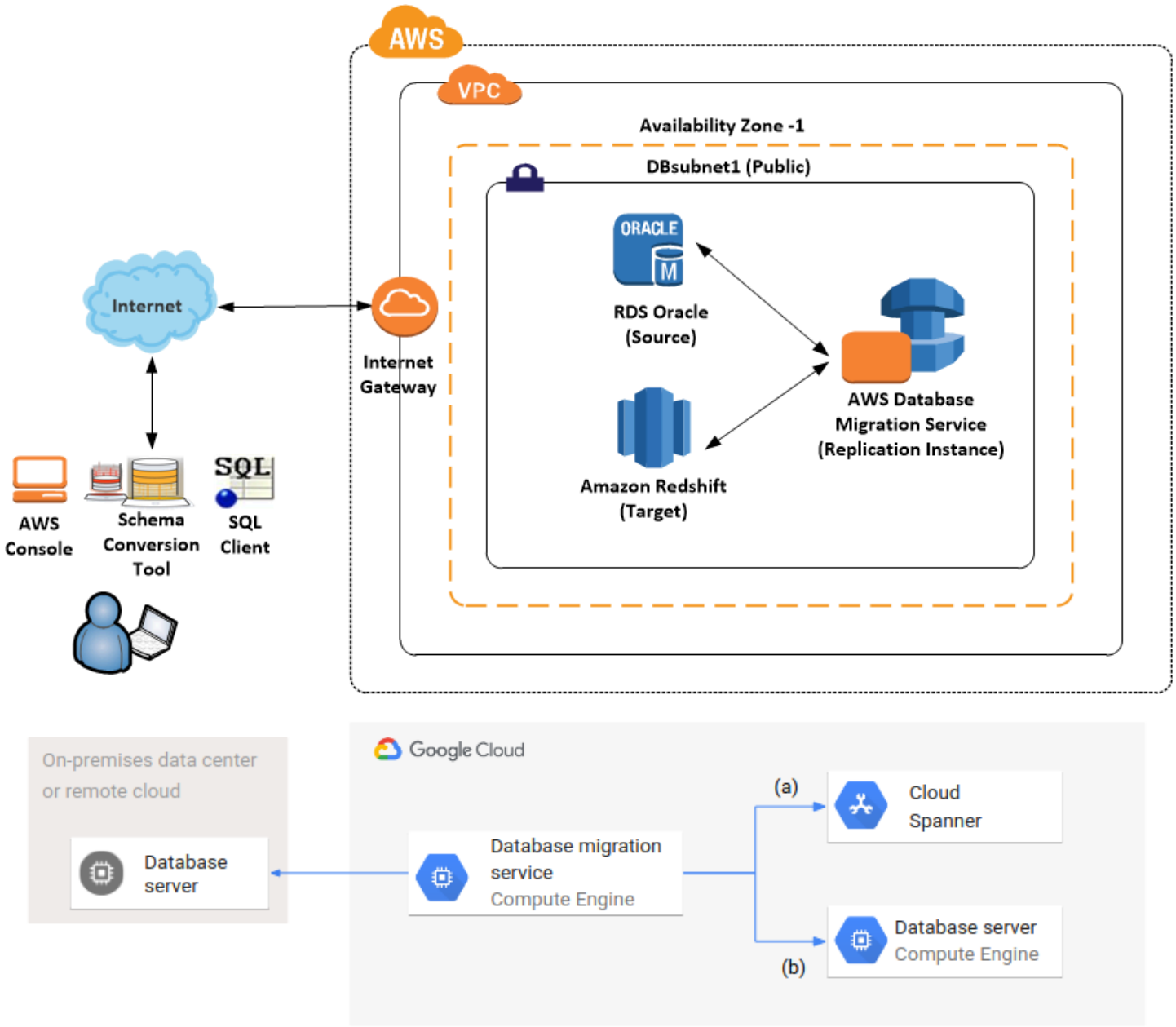
Data Migration

Cloud Data Migration Overview

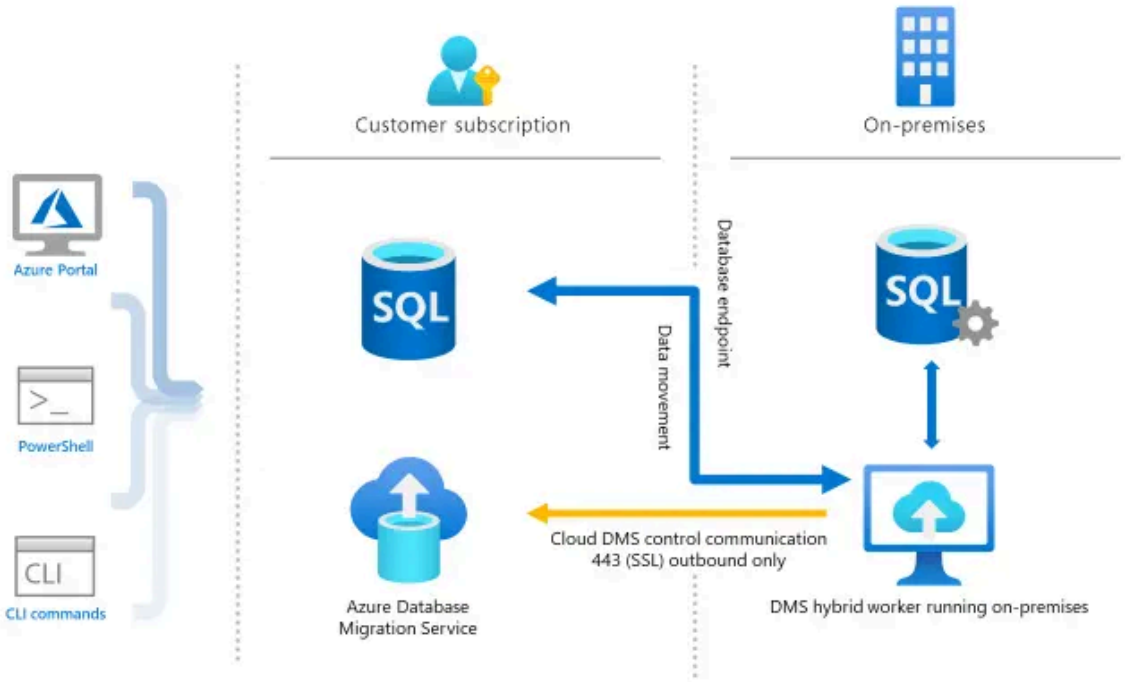
Cloud data migration is the process of transferring digital assets—such as data, applications, and services—from on-premises or legacy infrastructure to a cloud or between clouds. This can involve partial moves (just the data or selected services) or a comprehensive shift of all IT resources. The process benefits organizations by leveraging scalability, cost savings, reliability, and new analytics capabilities available in the cloud.

General Cloud Data Migration Steps

1. **Assessment & Planning**: **Analyze** data, applications, and **dependencies**; select a **migration strategy** (lift-and-shift, refactor, rebuild).
2. **Choosing Deployment Environment**: Decide on the **target cloud** provider (AWS, GCP, Azure), cloud type (public, private, hybrid), and **region**.
3. **Migration**: Execute migration using **tools**, **services**, and possibly **third-party software**. Migrate in phases, validate at each step.
4. **Testing & Validation**: **Verify data** and **applications** in the new environment using **test plans** and **data validation checks.**
5. **Optimization:** Refine cloud operations using **performance monitoring**, **resource management**, and **continuous improvement processes.**


AWS
VPC
Availability Zone -1
DBsubnet1 (Public)
ORACLE
RDS Oracle
(Source)
Internet
Internet
Gateway
AWS Database
Migration Service
(Replication Instance)
Amazon Redshift
(Target)
AWS
Console
Schema
Conversion
Tool
SQL
SQL
Client

On-premises data center
or remote cloud
Database
server
Google Cloud
Database migration
service
Compute Engine
(a)
Cloud
Spanner
Database server
Compute Engine
(b)



Data Migration Tools & Technologies

AWS (Amazon Web Services)

- **AWS Database Migration Service (DMS):** Migrates databases (homogeneous & heterogeneous) with minimal downtime. Supports live data migration, CDC, schema conversion with AWS Schema Conversion Tool (SCT).
- **AWS DataSync:** For fast and secure bulk data transfers (file or object storage) between on-premises and AWS (S3, EFS, FSx). Handles metadata, retry logic, and automation.
- **AWS Snow Family** (Snowcone, Snowball, Snowmobile): For very large-scale, offline data transfers when network-based transfers aren't practical.
- **Third-party tools**: **Integrate.io**, **Fivetran**, **Qlik** (for specific use cases or hybrid architectures).

GCP (Google Cloud Platform)

- **Cloud Storage Transfer Service**: For transferring large data sets from other clouds or on-premises to Google Cloud Storage.
- **Google Cloud Transfer Appliance**: Physical device for secure transfer of massive data sets when network bandwidth is limited.
- **Cloud Dataflow / Dataprep**: For in-cloud data transformation, validation, and workflow orchestration during migration.

Azure (Microsoft Azure)

- Azure Database Migration Service: For migrating on-premises or cloud databases (SQL, MySQL, PostgreSQL, Oracle, etc.) to Azure. Handles schema and data.
- Azure Data Factory: Visual orchestration platform for ETL, data integration, and large batch migrations.
- Azure Data Box: Hardware appliance to move very large data sets securely to Azure data centers.
- Azure Data Migration Assistant: Compatibility assessments and pre-migration checks.
- Azure Storage Migration Service: For storage/account migrations with integrity checks, encryption, support for various source platforms, and batch processing.

Best Practices for Cloud Data Migration

- **Thorough Assessment & Planning:** Inventory data and apps; understand dependencies and **data volumes**.
- **Select Migration Tools for Your Use Case:** Use managed services whenever possible (DMS for databases, **DataSync** for files, **Snowball** for large offline transfers, etc.).
- **Optimize Network & Transfer Performance**: Use dedicated links (**Direct Connect**/**ExpressRoute**), compress data, parallelize transfers for speed.
- **Data Security & Compliance**: **Encrypt** data in transit and at rest, maintain **audit trails/logging**, **validate** integrity using **checksums** or **hash values.**
- **Test & Validate:** Run pilot migrations to catch issues early, use validation tools, and build rollback/contingency plans.
- **Minimal Downtime:** Use CDC (Change Data Capture) and phased migration to keep production workloads running.
- **Post-Migration Optimization:** Monitor, adjust resources, and refine cloud environments for cost/performance.
- **Documentation:** Keep clear records of migration steps, tools used, and test results for compliance and knowledge transfer.

Step-by-Step AWS Data Migration Example

Let's walk through a typical scenario: migrating a MySQL on-premises database to Amazon RDS for MySQL using AWS DMS.

1. Preparation

- Assess the source database schema and size.
- Ensure network connectivity (VPN/Direct Connect between source and AWS).
- Set up an Amazon RDS MySQL instance as the migration target.

2. Schema Conversion

- Use AWS Schema Conversion Tool (SCT) to convert the source schema (if needed) for the RDS MySQL target.

- Apply the converted schema to the target database.

3. Set Up DMS Replication Instance

- Create a replication instance in AWS DMS to manage and perform the data transfer.

4. Configure Endpoints

- Source Endpoint: Enter the on-premises MySQL connection info.
- Target Endpoint: Enter Amazon RDS MySQL connection info.

5. Create a Migration Task

- Full Load: Migrate the existing data.
- Change Data Capture (CDC): Replicate ongoing changes during the migration.

6. Run & Monitor Migration

- Start the DMS migration task. Monitor with CloudWatch logs and the DMS console.
- Validate data integrity in the target database.

7. Cut Over

- Switch user and application access to the new AWS database.
- Decommission or repurpose on-premises resources when validated.

Sample AWS DMS Step-by-Step Code (AWS CLI)
Below is a sample approach (steps simplified) to create and run a DMS migration task using the AWS CLI:

```
# 1. Create a replication instance
        aws dms create-replication-instance \
          --replication-instance-identifier myreplinstance \
          --allocated-storage 100 \
          --replication-instance-class dms.r5.large \
          --engine-version 3.4.6

# 2. Create source and target endpoints

        aws dms create-endpoint \
          --endpoint-identifier src-mysql \
          --endpoint-type source \
          --engine-name mysql \
```

```
  --username admin \
  --password secretpw \
  --server-name YOUR_SRC_DB_HOST \
  --port 3306 \
  --database-name sourcedb

aws dms create-endpoint \
  --endpoint-identifier tgt-mysql \
  --endpoint-type target \
  --engine-name mysql \
  --username admin \
  --password secretpw \
  --server-name YOUR_RDS_ENDPOINT \
  --port 3306 \
  --database-name targetdb

# 3. Create the migration task
aws dms create-replication-task \
  --replication-task-identifier my-task \
  --source-endpoint-arn <src-endpoint-arn> \
  --target-endpoint-arn <tgt-endpoint-arn> \
  --migration-type full-load-and-cdc \
  --table-mappings file://mappings.json \
  --replication-task-settings file://task-settings.json

# 4. Start the migration task
aws dms start-replication-task \
  --replication-task-arn <task-arn> \
  --start-replication-task-type start-replication
```

Note: Replace placeholders with actual values and make sure you have the necessary IAM permissions and security groups configured. The table-mappings.json defines which tables to migrate and transformation rules, and task-settings.json contains task-level configuration.
Real-World Example: Migrating Files with AWS DataSync
If you want to migrate a file share or object storage (e.g., NFS server to S3) using DataSync:

1. Deploy the DataSync agent in your on-premises environment.
2. Configure source location (e.g., NFS mount).
3. Configure destination location (e.g., S3 bucket).
4. Create and run the DataSync task via AWS Console/CLI/Terraform.

Example (Terraform):
text
```
module "backup_tasks" {
  # ...module setup...
  datasync_tasks = [
    {
      name                 = "nfs_to_s3_task"
      source_location_arn     = "<nfs-location-arn>"
      destination_location_arn= "<s3-location-arn>"
      schedule_expression     = "rate(1 hour)" # Runs every hour
      # more options...
    }
  ]
}
```
This regularly syncs your files with integrity checks, retries, and logging out-of-the-box.
Key Takeaways

- Choose migration tools and services based on data type, source/target, scale, and downtime requirements.
- Use built-in features for security, reliability, and automation.
- Follow a structured methodology for planning, execution, and validation.
- Test, monitor, and optimize every stage of your migration.

AWS, GCP, and Azure all provide robust migration platforms, managed services, and best practices to reduce risk and complexity in your journey to the cloud.