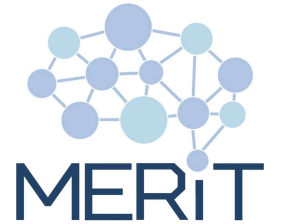




Co-funded by
the European Union



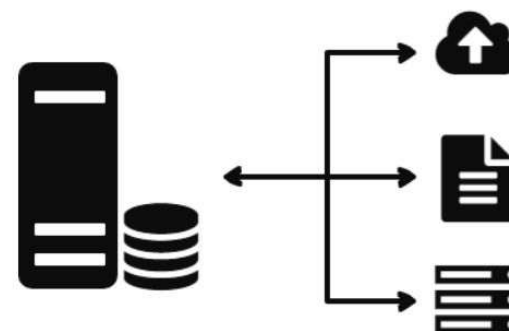
Database Security

Backup and Disaster Recovery

Why Backup & Restore



- The primary reason for data backup is to save important files in the event of a system crash or hard drive failure. Additional data backups should be made if the original backups result in data corruption or hard drive failure. This is best accomplished through the use of the cloud or offsite storage.



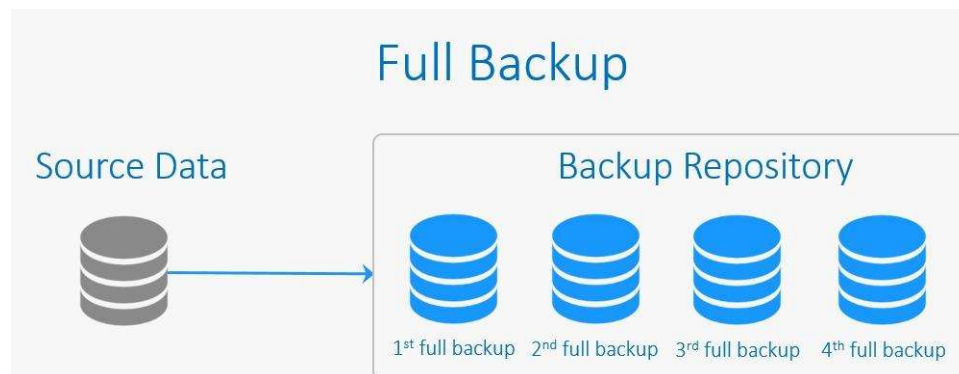
Types of Backup

- ☐ Full backup
- ☐ Incremental backup
- ☐ Differential backup



What is a full backup?

□ In a single backup operation, a full backup is the process of creating at least one additional copy of all data files that an organization wishes to protect. A backup administrator or other data protection specialist designates which files are duplicated during the full backup process in advance. Typically, the data copied includes application files, some metadata that helps make the files more useful to the applications, logs and tracking files that document what the applications are processing, and other control and management files.

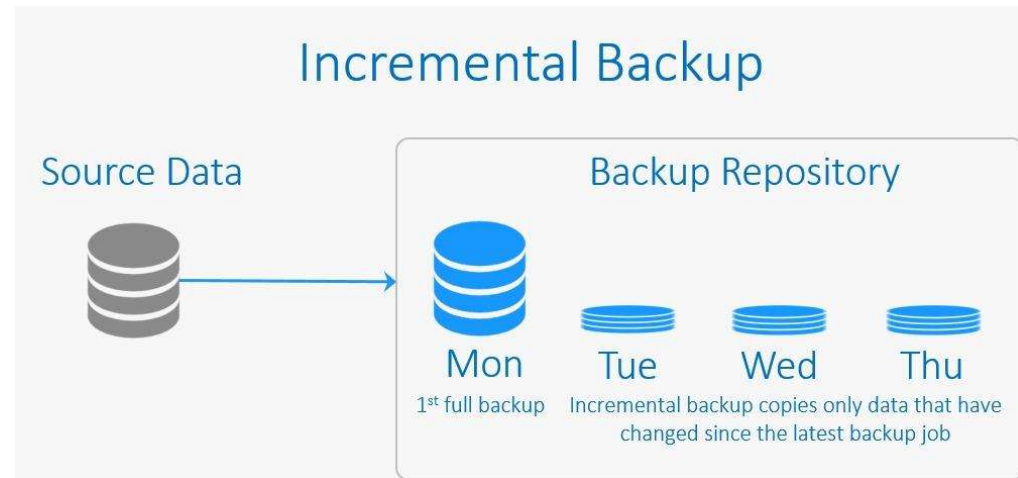
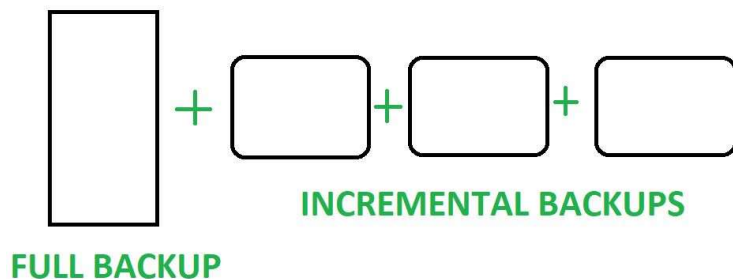


A Full Backup Example

The entire directory of folders and files on that drive will be copied on Tuesday. On Friday, any new files added to that drive will be copied, as will the entire directory from before.

What is a Incremental Backup?

- An incremental backup is a type of backup that only copies data that has changed or been created since the previous backup activity. When the amount of data to be protected is too large to do a full backup of that data every day, an incremental backup approach is used.

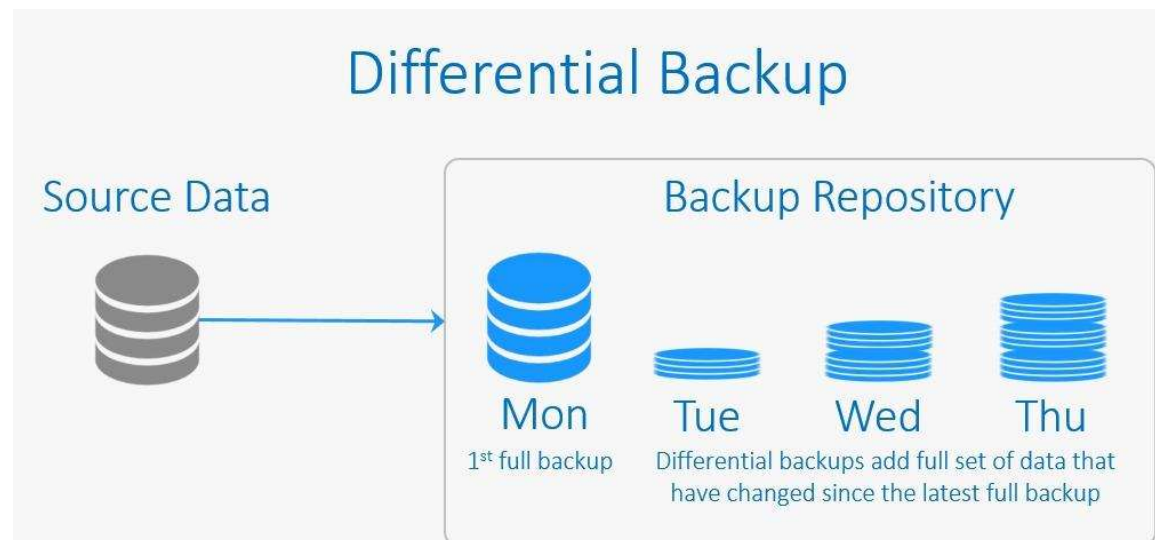


Example of incremental backup

An incremental backup scenario necessitates one full backup followed by incremental backups over time. For example, if a full backup was performed on Monday, the incremental backup on Tuesday will take a snapshot and backup all new or changed files since the previous backup.

What is a Differential backup?

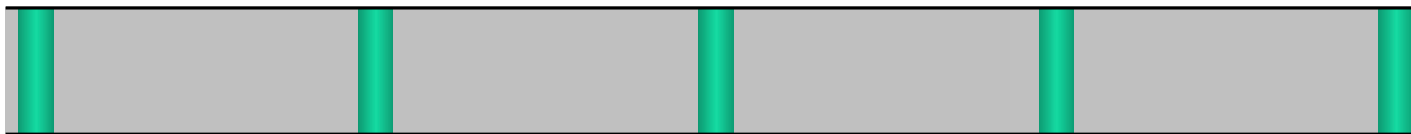
- A differential backup is a data backup that copies all of the files that have changed since the last full backup was performed. This includes any data that has been created, updated or altered in any way and does not copy all of the data every time.



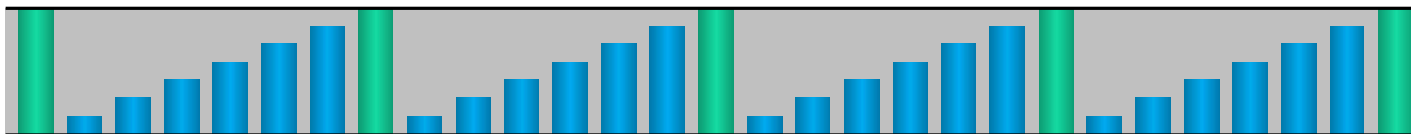
For example, if a full backup was performed on Monday, the incremental backup on Tuesday will include all new or changed files since Monday's backup. However, the incremental backup on Wednesday will only backup files that have changed since the previous incremental backup on Tuesday, and so on until another full backup is performed.

Backup Granularity and Levels

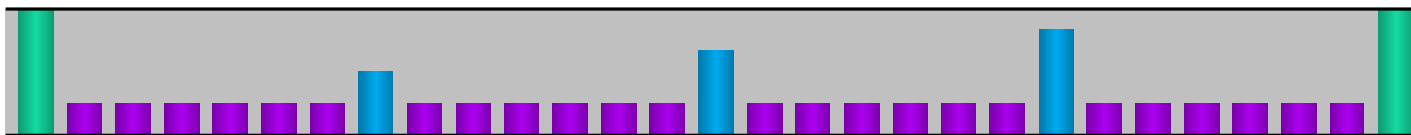
Full Backup



Cumulative (Differential)



Incremental



Full



Cumulative



Incremental

All Types of Backup functions

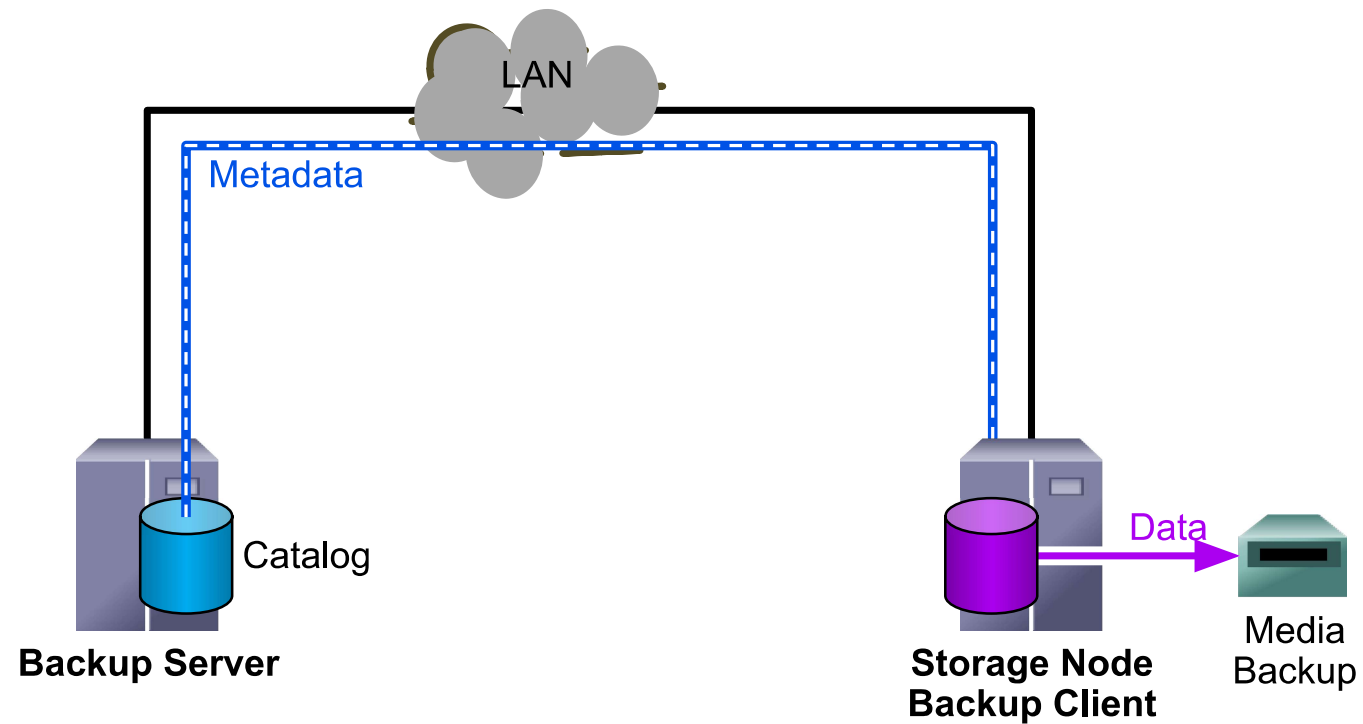
□ In terms of backup time (and thus backup speed), a full backup takes longer than a differential or incremental backup. The fastest backup method is incremental backup, followed by differential backup.

	full backup	Incremental backup	Differential backup
 storage space	High	Medium to High	Low
backup speed	Slow	Fast	Fastest
Restoration Speed	Fastest	Fast	Slow
Media Required for recovery	Most recent Backup only	Most recent full backup & most recent differential backup	Most recent full backup & most recent differential backup

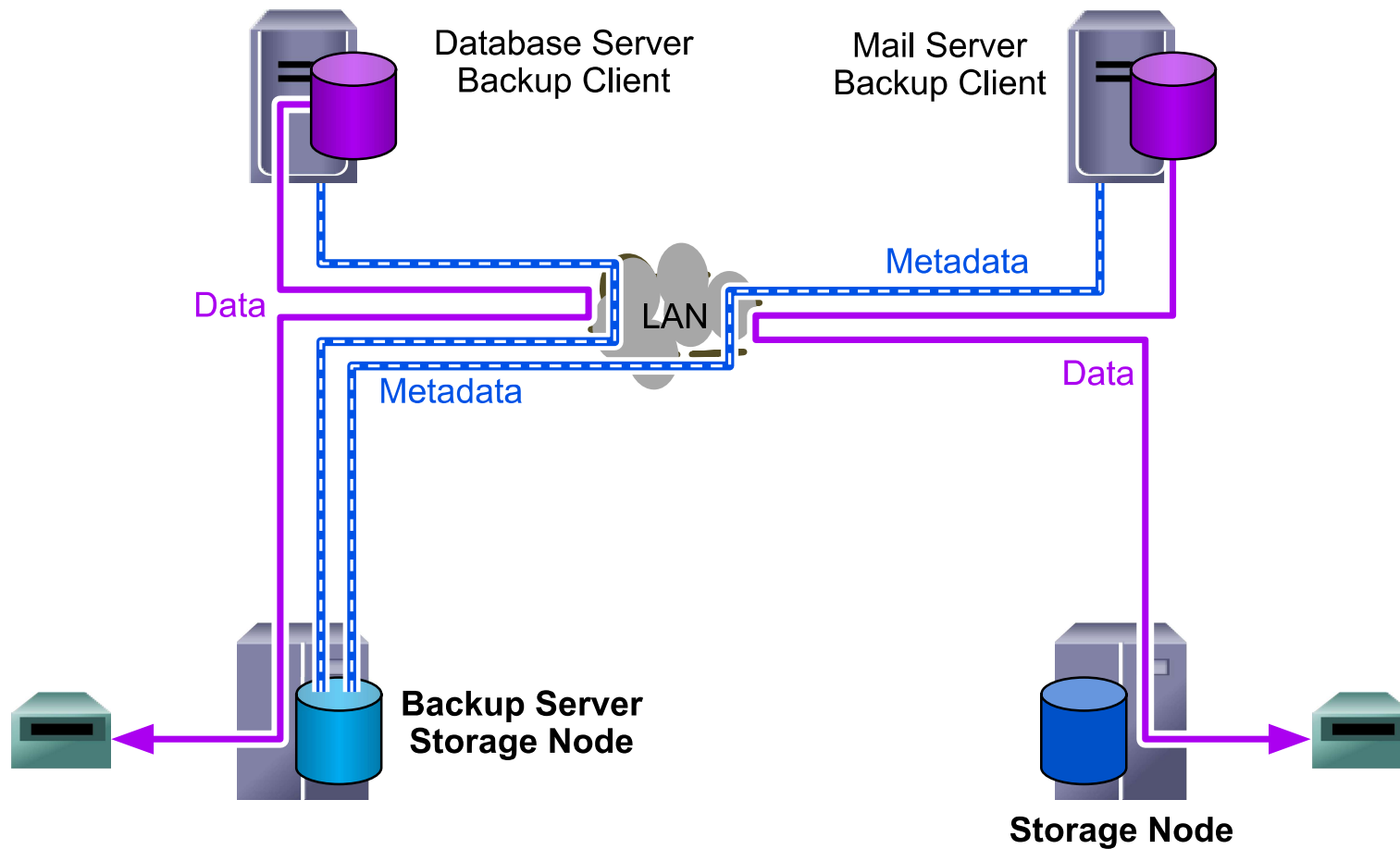
Backup Architecture Topologies

- ☐ There are 3 basic backup topologies:
 - ✓ Direct Attached Based Backup
 - ✓ LAN Based Backup
 - ✓ SAN Based Backup
- ☐ These topologies can be integrated, forming a “mixed” topology

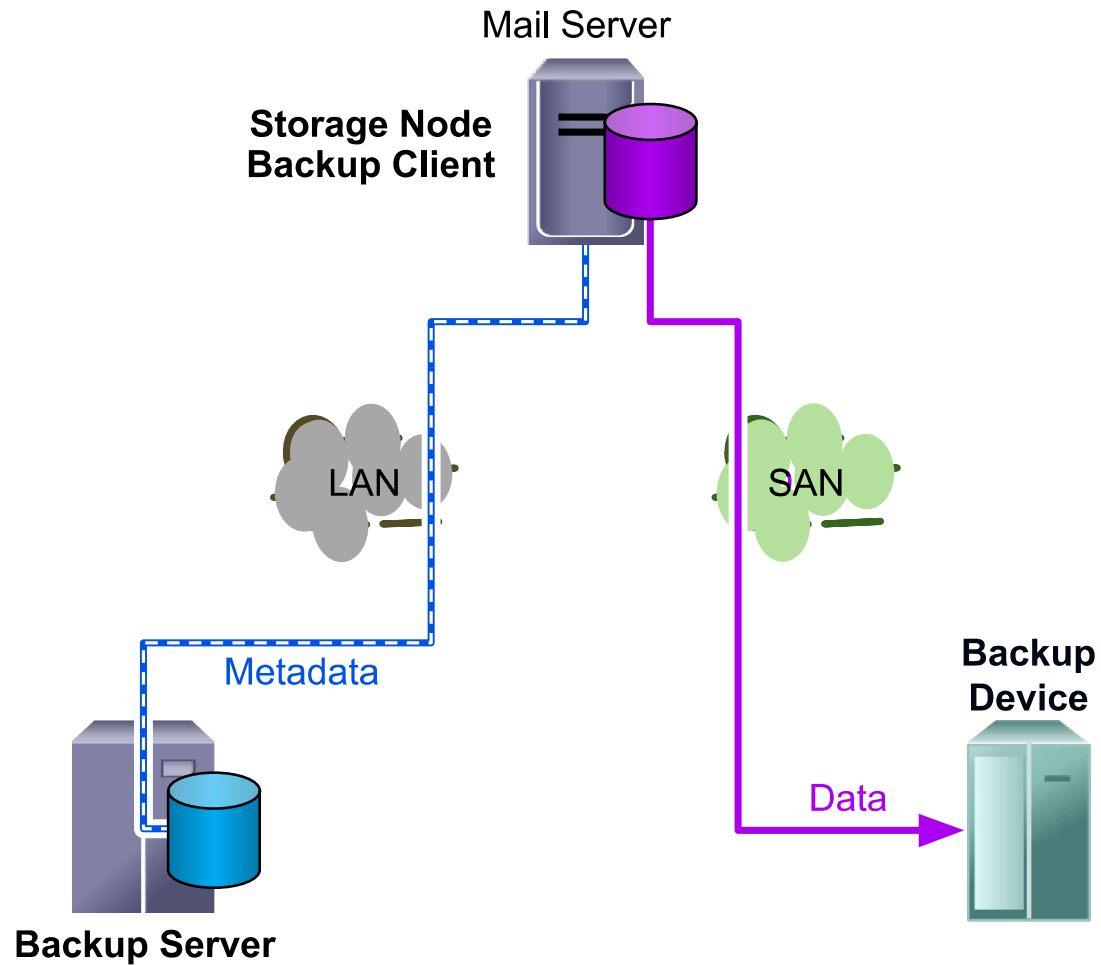
Direct Attached Based Backups



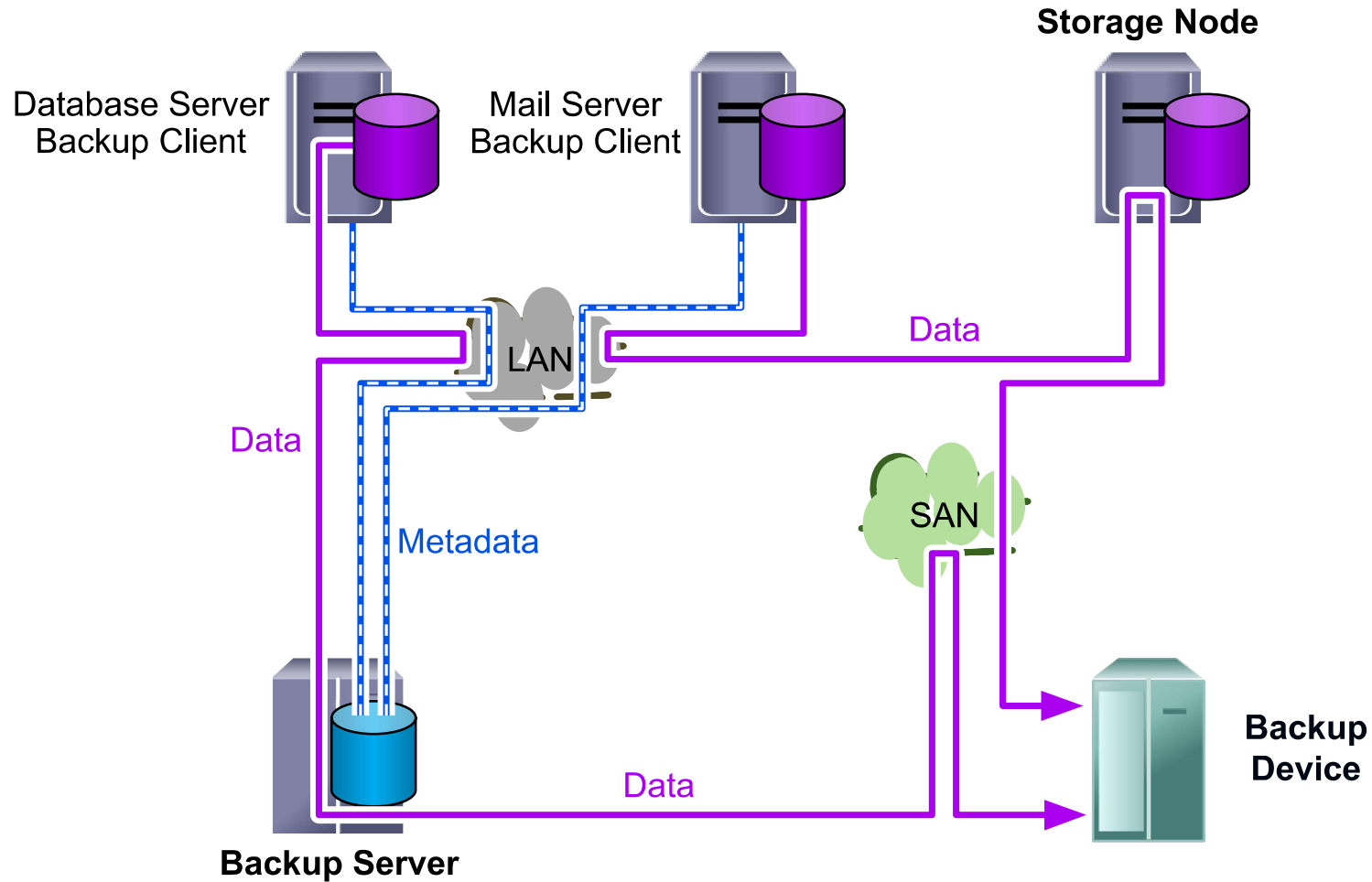
LAN Based Backups



SAN Based Backups (LAN Free)



SAN/LAN Mixed Based Backups



Automated Backup Strategies – General Concept

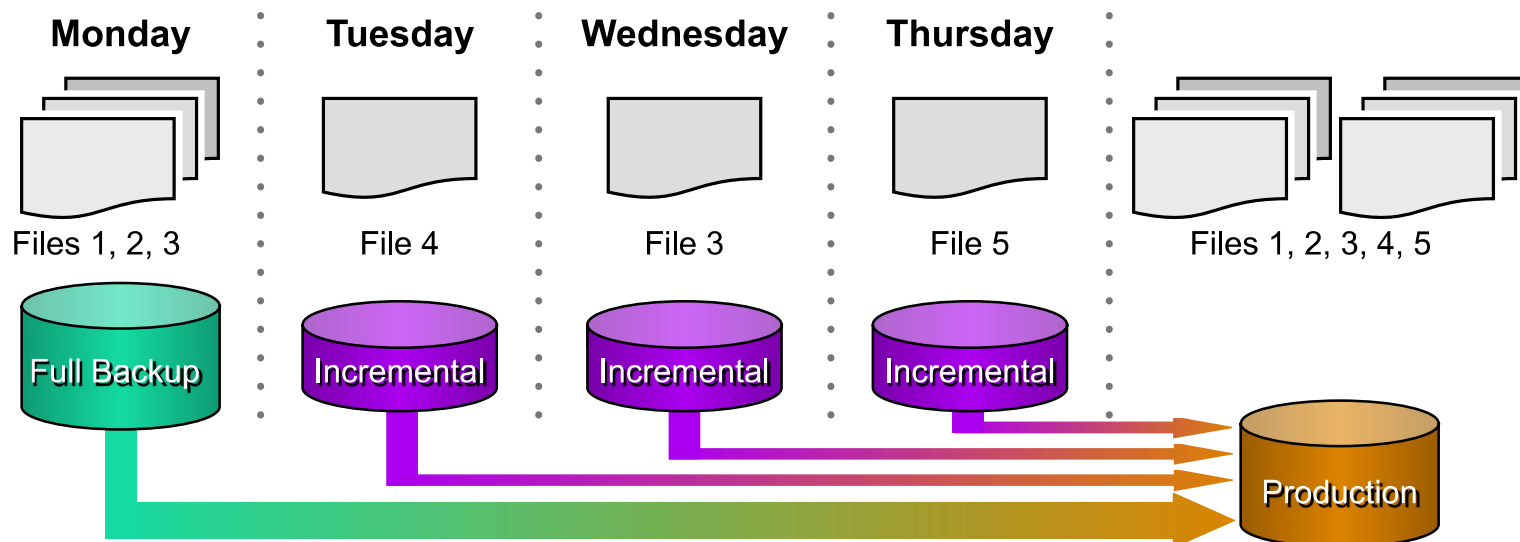
What are they?

Automated backup strategies use tools and technologies to manage backups regularly, securely, and without manual intervention.

Key objectives:

- ✓ Minimize human error (mistakes, misconfigurations)
- ✓ Ensure business continuity with reliable restore points
- ✓ Improve operational efficiency by freeing IT resources

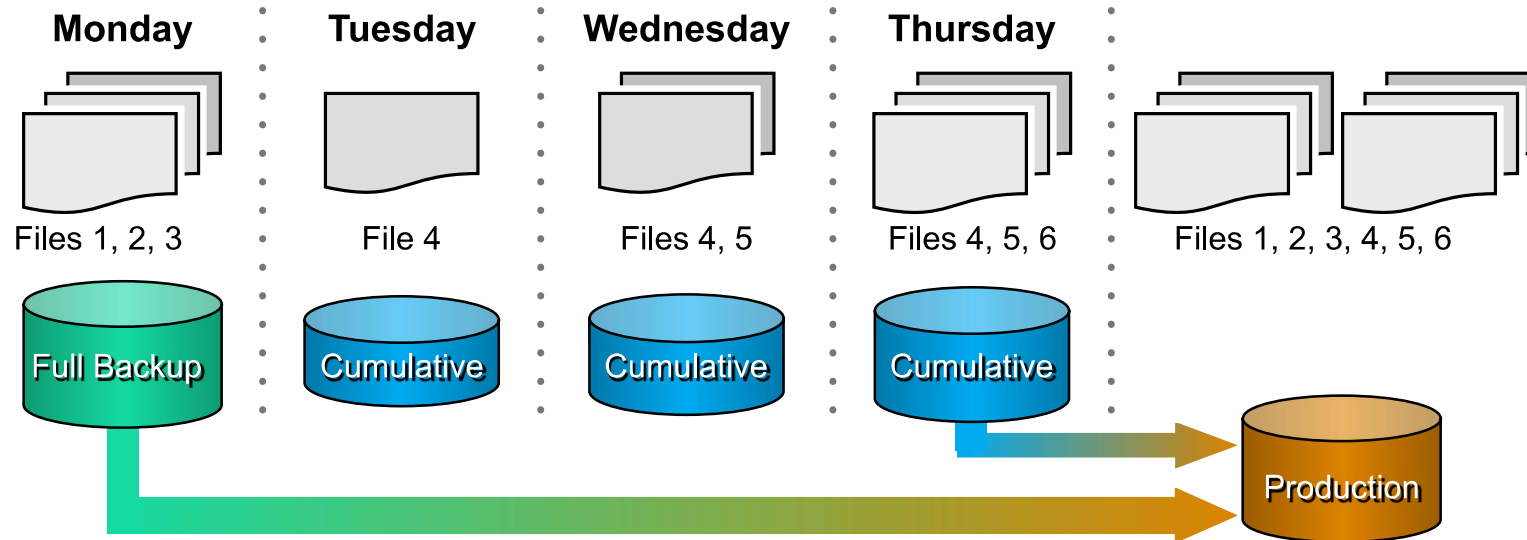
Restoring an Incremental Backup



Key Features

- ✓ Files that have changed since the last full or incremental backup are backed up.
- ✓ Fewest amount of files to be backed up, therefore faster backup and less storage space.
- ✓ Longer restore because last full and all subsequent incremental backups must be applied.

Restoring a Differential (Cumulative) Backup



Key Features

- ✓ More files to be backed up, therefore it takes more time to backup and uses more storage space.
- ✓ Much faster restore because only the last full and the last cumulative backup must be applied.

Components of an Automated Backup Strategy



1. Scheduled Backups

- Daily, weekly, or monthly intervals depending on data criticality.
- Task schedulers like `cron` or built-in DBMS tools.

2. Scripts & Retention Policies

- Custom scripts for selecting data, compressing, encrypting.
- Data retention rules (e.g., last 30 days only).

3. Notifications & Alerts

- Email alerts or dashboards for backup status.
- Logging for auditing and troubleshooting.

Tools and Real-World Examples



On-premise Tools:

- `cron + rsync`, `mysqldump`, `pg\_dump`, Veeam, Duplicati

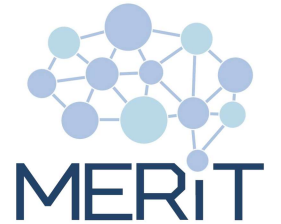
Cloud Services:

- AWS Backup, Azure Backup, Google Cloud Backup & DR

Advantages of Cloud Backups:

- Scalability and high availability.
- Geographical redundancy.
- Compliance (ISO 27001, GDPR).

Script 1: File Backup Using rsync and cron



```
#!/bin/bash
# Backup script using rsync
# Author: Rubén López Barrio

# Set source and destination directories
SOURCE="/home/user/documents"
DEST="/mnt/backup_drive/documents_backup"

# Get current date for backup folder name
DATE=$(date +%Y-%m-%d)

# Create a dated backup folder
mkdir -p "$DEST/$DATE"

# Perform rsync backup (preserve permissions, compress during transfer, show progress)
rsync -avz --progress "$SOURCE/" "$DEST/$DATE/"

# Optional: Log the backup process
echo "Backup completed on $DATE" >> /var/log/backup.log
```



Automate with cron (run every night at 1:00 AM):

```
0 1 * * * /path/to/rsync_backup.sh
```

Script 2: MySQL Database Backup Using mysqldump



```
#!/bin/bash
# MySQL database backup script
# Author: Rubén López Barrio

# Define DB credentials and backup path
USER="backupuser"
PASSWORD="securepassword"
DATABASE="companydb"
BACKUP_DIR="/mnt/backup_drive/db_backups"
DATE=$(date +%F_%H-%M)

# Create backup directory if it doesn't exist
mkdir -p "$BACKUP_DIR"

# Dump the database into a compressed .sql.gz file
mysqldump -u $USER -p$PASSWORD $DATABASE | gzip >
"$BACKUP_DIR/${DATABASE}_${DATE}.sql.gz"

# Log the operation
echo "MySQL backup for $DATABASE created on $DATE" >> /var/log/db_backup.log
```

 Automate with cron (run every day at 2:00 AM):

```
0 2 * * * /path/to/mysql_backup.sh
```

Script 3: Retention Policy (Delete Old Backups)



```
#!/bin/bash
# Retention policy script
# Deletes backups older than 30 days
# Author: Rubén López Barrio

# Backup directory to clean
BACKUP_DIR="/mnt/backup_drive/documents_backup"

# Find and delete backup folders older than 30 days
find "$BACKUP_DIR" -type d -mtime +30 -exec rm -rf {} \;

# Log the cleanup
echo "Retention cleanup run on $(date)" >>
/var/log/retention_cleanup.log
```

 **Automate with cron (run weekly on Mondays at 3:00 AM):**

```
0 3 * * 1 /path/to/retention_policy.sh
```

Secure Storage and Restoration Processes



These are security measures applied during the storage and recovery of backups to prevent data loss or unauthorized access.

They ensure data confidentiality, availability, and integrity across all backup stages.

The process is divided into two key areas: **secure storage and secure restoration.**

Secure Storage – Key Measures



- ✓ Encrypt backups both in transit and at rest (e.g., using AES-256).
- ✓ Use external or cloud storage with strict access controls.
- ✓ Apply geographic redundancy (store backups in different locations to avoid loss due to disasters).
- ✓ Restrict backup access to authorized personnel only.
- ✓ Regularly verify backup integrity using hashes or checksums.

Script 1: Secure Storage – Encrypted and Remote Backup



```
#!/bin/bash
# Secure Backup Script: Encrypt and Upload to Remote Server
# Author: Rubén López Barrio

# Variables
SOURCE="/home/user/documents"
BACKUP_DIR="/tmp/secure_backup"
BACKUP_NAME="backup_$(date +%F).tar.gz"
ENCRYPTED_NAME="$BACKUP_NAME.gpg"
REMOTE_USER="backupuser"
REMOTE_HOST="backup.example.com"
REMOTE_DIR="/home/backupuser/backups"

# Create backup directory
mkdir -p "$BACKUP_DIR"

# Compress source folder
tar -czf "$BACKUP_DIR/$BACKUP_NAME" -C "$SOURCE" .

# Encrypt the backup using GPG with AES256
gpg --batch --yes --symmetric --cipher-algo AES256 --passphrase "StrongPass123" "$BACKUP_DIR/$BACKUP_NAME"

# Secure copy (scp) to remote storage with control access
scp "$BACKUP_DIR/$ENCRYPTED_NAME" "$REMOTE_USER@$REMOTE_HOST:$REMOTE_DIR"

# Clean up temporary files
rm -rf "$BACKUP_DIR"

# Log the operation
echo "Encrypted backup sent to $REMOTE_HOST on $(date)" >> /var/log/secure_backup.log
```

Secure Restoration – Best Practices



- ✓ Validate the integrity of the backup before restoration.
- ✓ Ensure authentication and access control during the restoration process.
- ✓ Perform restoration in secure, controlled environments.
- ✓ Maintain logs and audit trails of all restoration operations.
- ✓ Delete decrypted or temporary files immediately after use.



Script 2: Secure Restoration – Decrypt and Restore with Integrity Check

```
#!/bin/bash
# Secure Restore Script: Decrypt and Load Data
# Author: Rubén López Barrio

# Variables
BACKUP_FILE="/home/backupuser/backups/backup_2025-04-15.tar.gz.gpg"
DECRYPTED_FILE="/tmp/decrypted_backup.tar.gz"
RESTORE_DIR="/home/user/restore_target"
CHECKSUM_EXPECTED="2b8f403bd947...yoursha256...abcd"

# Decrypt the backup
gpg --batch --yes --passphrase "StrongPass123" -o "$DECRYPTED_FILE" --decrypt "$BACKUP_FILE"

# Verify file integrity
CHECKSUM_ACTUAL=$(sha256sum "$DECRYPTED_FILE" | awk '{print $1}')

if [ "$CHECKSUM_ACTUAL" != "$CHECKSUM_EXPECTED" ]; then
    echo "Integrity check failed. Aborting restoration." >> /var/log/restore.log
    exit 1
fi

# Extract backup to restore location
mkdir -p "$RESTORE_DIR"
tar -xzf "$DECRYPTED_FILE" -C "$RESTORE_DIR"

# Log restoration
echo "Backup restored successfully on $(date)" >> /var/log/restore.log

# Clean up sensitive temporary files
rm -f "$DECRYPTED_FILE"
```

Summary



- **Purpose:** Prevent data loss from failures or disasters.
 - **Backup Types:**
 - *Full:* All data.
 - *Incremental:* Only changes since last backup.
 - *Differential:* Changes since last full backup.
 - **Topologies:**
 - DAS, LAN-based, SAN-based, or mixed.
 - **Automation:**
 - Scheduled tasks, scripts, alerts, retention policies.
 - **Security:**
 - Encryption, access control, redundancy, integrity checks.
 - **Tools:**
 - On-premise: rsync, mysqldump.
 - Cloud: AWS, Azure, GCP.
- Goal:** Ensure reliable backup and fast, secure recovery.