Understanding NFS Service in Linux

How It Works and Configuration Essentials

A comprehensive guide to implementing and optimising Network File System services for seamless file sharing across Linux environments.

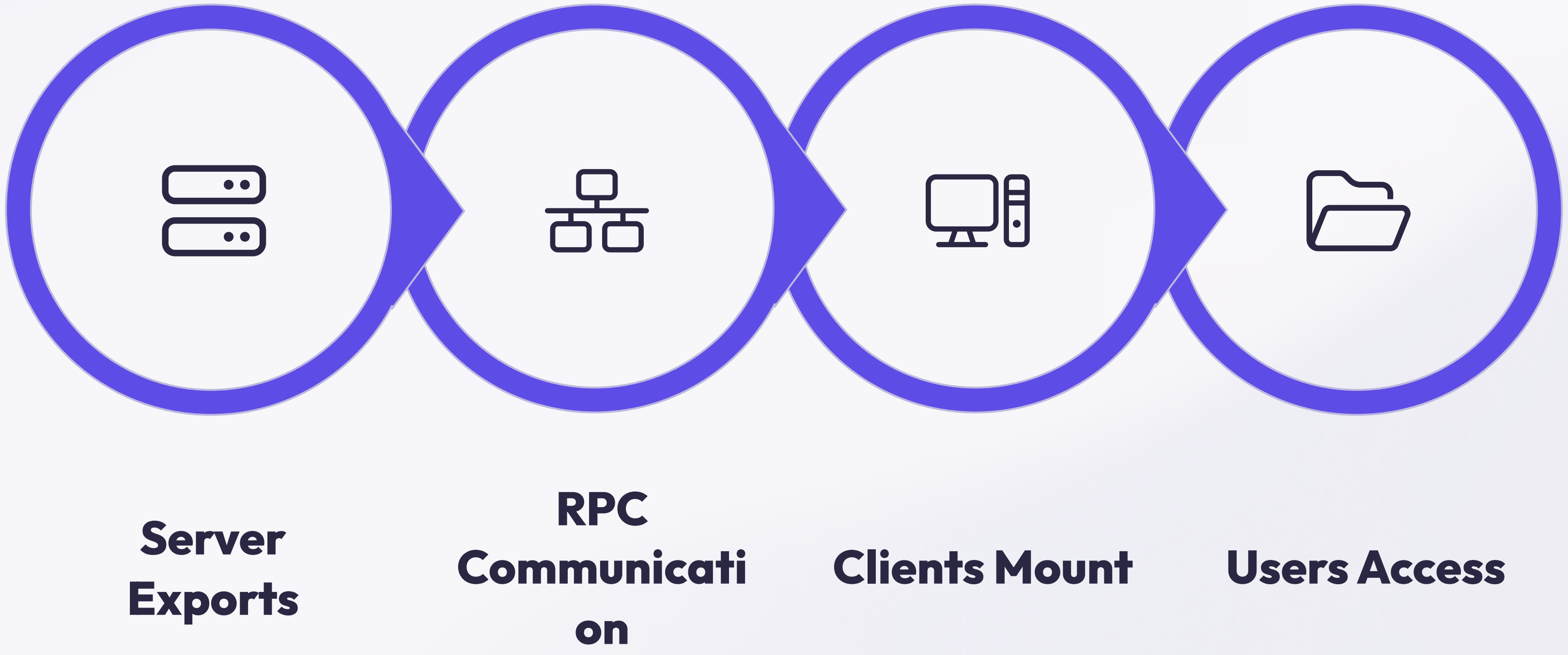
What is NFS?

Network File System (NFS) allows Linux systems to share directories and files over a network as if they were local storage devices. Originally developed by Sun Microsystems in 1984, it has evolved to version 4.2 with significantly enhanced features.

Key Benefits

- Centralised storage access across multiple systems
- Reduction in disk space duplication
- Simplified data management and backup processes
- Transparent file access for users and applications

How NFS Works: The Basics



NFS relies on Remote Procedure Calls (RPC) to facilitate communication between servers and clients. Whilst earlier versions required multiple ports, NFSv4 simplifies firewall traversal by using a single port (2049), making it significantly more secure and easier to implement in restricted network environments.

Key Ports Used by NFS

Port 2049 TCP/UDP

Main NFS daemon port (nfsd). This is the primary service port for all NFS communication, especially in NFSv4.

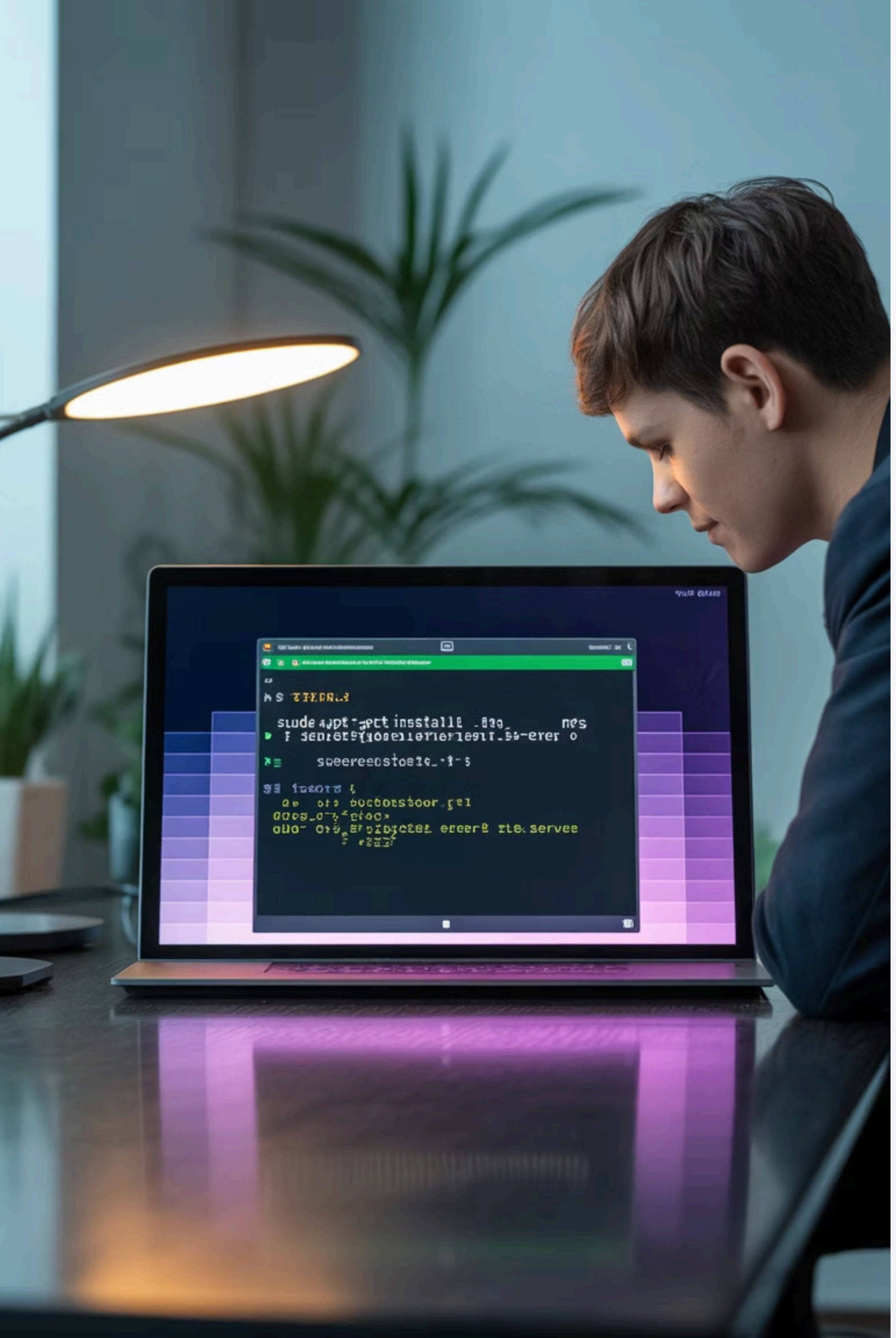
Port 111 TCP/UDP

Portmapper (rpcbind) service for mapping RPC services to dynamic ports. Essential for NFSv3 and earlier.

Port 20048 TCP/UDP

Default mountd port, with additional services like lockd and statd potentially using dynamic or fixed ports.

Best practice: Configure mountd and lockd to use fixed ports in your NFS configuration to avoid firewall traversal issues and improve security.



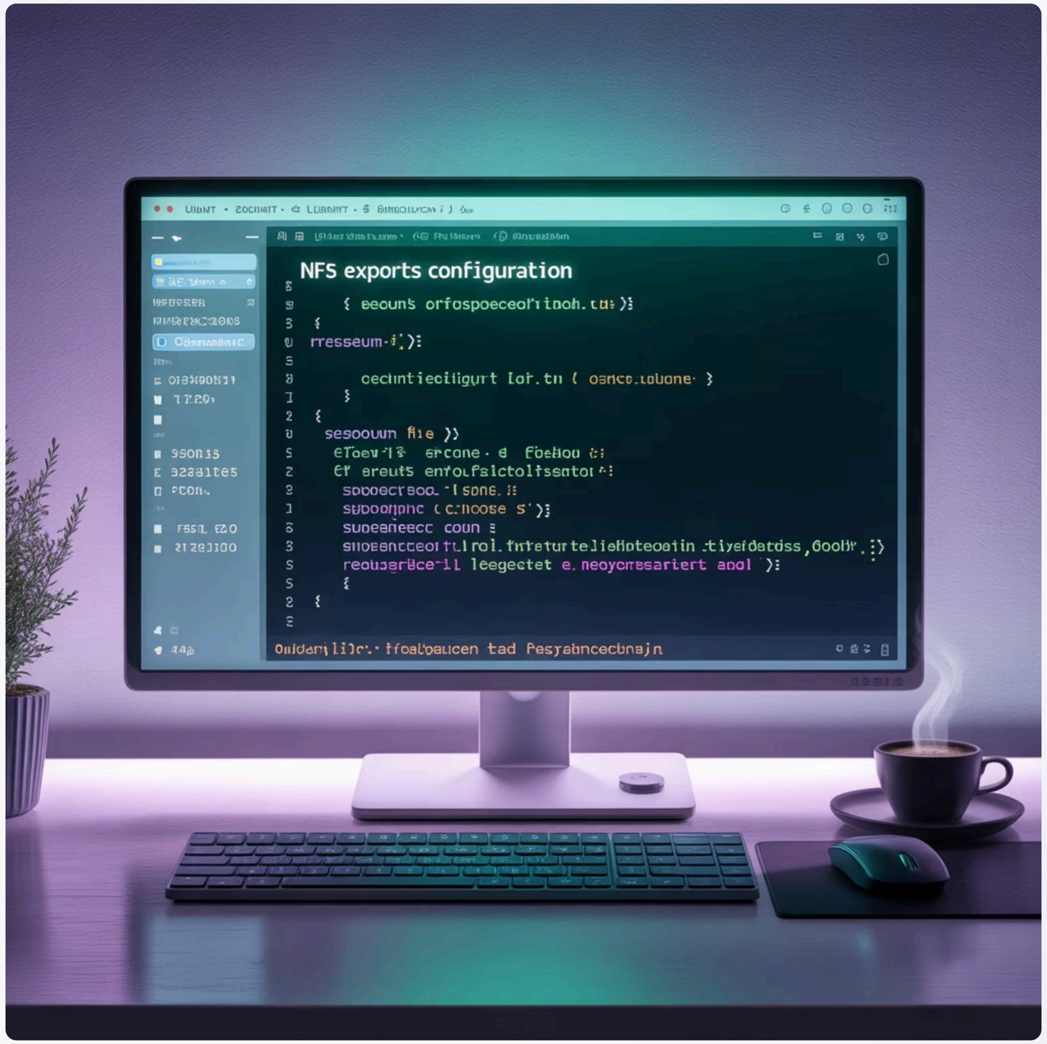
Installing NFS on Linux

(Example: Ubuntu/Debian)

```
# On the server
sudo apt update && sudo apt install nfs-kernel-server

# On the client
sudo apt update && sudo apt install nfs-common

# Enable and start NFS server
sudo systemctl enable --now nfs-server

# Start rpcbind service
sudo systemctl enable --now rpcbind
```

After installation, verify the service status with systemctl status nfs-server to ensure proper functionality before proceeding with configuration.

Configuring NFS Shares: /etc/exports File



The /etc/exports file is the central configuration file where you define which directories are shared and who can access them.

Example Configurations

```
# Share with everyone, read-write
/srv/nfs *(rw,sync,no_subtree_check)

# Share with specific subnet
/nfs/exports/myshare 192.168.122.0/24(rw,sync)

# Share with specific host, read-only
/opt/readonly client.example.com(ro,sync)
```

After modifying the exports file, apply changes with:

```
sudo exportfs -r
```

Common NFS Share Options Explained

Access Control

- `ro`: Read-only access
- `rw`: Read-write access

Write Behaviour

- `sync`: Safer, confirms writes before returning
- `async`: Faster but riskier during power failures

User Mapping

- `all_squash`: Maps all users to anonymous
- `root_squash`: Maps root to anonymous (default)
- `no_root_squash`: Allows root access (caution!)

These options provide granular control over how clients interact with your shared filesystems, balancing security with accessibility requirements.

Mounting NFS Shares on Client

Create Mount Point

```
sudo mkdir -p /mnt/myshare
```

Establish a directory where the remote filesystem will be accessible.

Mount Manually

```
sudo mount -t nfs
server_ip:/nfs/exports/myshare
/mnt/myshare
```

Connect to the NFS share immediately for current session use.

Mount Automatically at Boot

```
server_ip:/nfs/exports/myshare
/mnt/myshare nfs defaults 0 0
```

Add this line to /etc/fstab for persistent mounting after system restarts.

Verify successful mounting with `df -h` or `mount | grep nfs` to see active NFS connections.

Security and Performance Tips

Port Configuration

Fix dynamic ports (mountd, lockd) in `/etc/nfs.conf` or `/etc/sysconfig/nfs` to simplify firewall rules:

```
[mountd]
port=8649
```

Data Safety

Use `sync` option for critical data to prevent corruption during system failures, despite potential performance impact.

Access Restriction

Limit client access by IP or hostname rather than using wildcards:

```
/exports/data 192.168.1.0/24(rw,sync)
```

Regular Auditing

Check ownership and permissions on shared directories regularly to prevent unintended access or privilege escalation.

Consider implementing NFSv4 with Kerberos for enhanced security in production environments with sensitive data.

Summary: Mastering NFS for Efficient Linux File Sharing

Setup

NFS provides seamless network file sharing with flexible configuration options across Linux environments.

Security

Understanding ports and share options is key to establishing a secure, reliable setup with appropriate access controls.

Operation

Proper installation, export configuration, and client mounting ensure smooth operational performance for shared resources.

With these best practices implemented, NFS becomes a powerful tool for centralised storage management in Linux environments, providing both flexibility and efficiency for system administrators.