Table of Contents

Part 2 of 2 - Explore all 232+ Linux commands at dargslan.com/learn/linux-commands
Each command includes syntax, options, practical examples with output, and pro tips.

$ nft

Advanced

nftables command line tool (modern firewall)

nft is the command-line tool for nftables, the modern replacement for iptables. nftables provides unified IPv4/IPv6 packet filtering, NAT, and classification with a cleaner, more consistent syntax.

nftables is the default firewall framework in newer Linux distributions (Debian 10+, RHEL 8+). It ...

Options & Flags

`list ruleset`	Show all rules
`add table`	Create a table
`add chain`	Create a chain
`add rule`	Add a rule
`flush ruleset`	Clear all rules
`delete rule`	Remove a rule by handle

Practical Examples

Example: List all rules

```
$ sudo nft list ruleset
```

Shows the complete firewall ruleset.

Example: Allow SSH and web

```
$ sudo nft add rule inet filter input tcp dport { 22, 80, 443 } accept
```

Allows SSH, HTTP, and HTTPS using a set.

Example: Block IP

```
$ sudo nft add rule inet filter input ip saddr 192.168.1.100 drop
```

Drops all traffic from a specific IP.

Example: Save rules

```
$ sudo nft list ruleset > /etc/nftables.conf
```

Saves the current ruleset to a file for persistence.

Example: Load rules

```
$ sudo nft -f /etc/nftables.conf
```

Loads rules from a file.

Tips & Best Practices

Note: nftables replaces iptables: nftables is the modern successor. Most distributions still support iptables through compatibility layers, but new setups should use nft.

Pro Tip: Sets for efficiency: nft supports sets: tcp dport { 80, 443, 8080 } accept. More efficient than multiple rules.

Warning: Rules are not persistent by default: Save with nft list ruleset > /etc/nftables.conf. Enable nftables.service for automatic loading on boot.

$ openssl

Advanced

OpenSSL command line tool for cryptography and SSL/TLS

openssl is a cryptographic toolkit for SSL/TLS operations, certificate management, encryption, hashing, and key generation. It is the Swiss Army knife of cryptography on Linux.

openssl is used for generating SSL certificates, creating certificate signing requests (CSRs), testing SSL connections,...

Options & Flags

`genrsa`	Generate RSA private key
`req`	Create certificate signing request
`x509`	Create/examine X.509 certificates
`s_client`	Test SSL/TLS connection
`enc`	Encrypt/decrypt files
`rand`	Generate random data
`dgst`	Calculate hash/digest

Practical Examples

Example: Generate private key

```
$ openssl genrsa -out server.key 4096
Generating RSA private key, 4096 bit long modulus
```

Creates a 4096-bit RSA private key.

Example: Generate CSR

```
$ openssl req -new -key server.key -out server.csr
```

Creates a Certificate Signing Request for a CA.

Example: Self-signed certificate

```
$ openssl req -x509 -newkey rsa:4096 -keyout key.pem -out cert.pem -days 365 -nodes
```

Creates a self-signed certificate valid for 1 year.

Example: Test SSL connection

```
$ openssl s_client -connect example.com:443 -servername example.com
```

Tests SSL/TLS connection and shows certificate details.

Example: View certificate

```
$ openssl x509 -in cert.pem -text -noout
```

Displays certificate details: issuer, validity, subject, etc.

Tips & Best Practices

Pro Tip: Check remote certificates: echo | openssl s_client -connect host:443 2>/dev/null | openssl x509 -text quickly shows any server certificate.

Warning: Protect private keys: Private keys must be 600 permissions and never shared. chmod 600 server.key immediately after generation.

Note: Let's Encrypt: For production SSL certificates, use certbot (Let's Encrypt) instead of self-signed. openssl is for testing and CSR generation.

$ sha256sum

Beginner

Compute and verify SHA-256 message digests

sha256sum calculates and verifies SHA-256 cryptographic hashes (256-bit). SHA-256 is part of the SHA-2 family and is currently the recommended standard for file integrity verification and security.

sha256sum produces a unique 64-character hexadecimal fingerprint. It is computationally infeasible...

Options & Flags

`file`	Calculate SHA-256 hash
`-c`	Check hashes from file
`-b`	Binary mode
`--quiet`	Only show failures

Practical Examples

Example: Calculate hash

```
$ sha256sum ubuntu.iso
a1b2c3d4e5f6... ubuntu.iso
```

Computes the SHA-256 hash of a file.

Example: Verify download

```
$ sha256sum -c SHA256SUMS
ubuntu.iso: OK
```

Checks files against their expected SHA-256 hashes.

Example: Hash a string

```
$ echo -n "password123" | sha256sum
ef92b778bafe... -
```

Calculates SHA-256 of a string.

Example: Create checksums

```
$ sha256sum *.tar.gz > SHA256SUMS
```

Generates a checksum file for all archives.

Example: Multiple files

```
$ sha256sum backup_*.tar.gz
```

Calculates hashes for all backup files.

Tips & Best Practices

Pro Tip: Always use SHA-256 over MD5: SHA-256 is the current standard for file verification. MD5 is broken for security. sha256sum should be your default.

Note: Verify Linux downloads: Linux distributions provide SHA256SUMS files. Download both the ISO and SHA256SUMS, then run sha256sum -c SHA256SUMS.

Warning: Use -n with echo: echo "text" | sha256sum includes a trailing newline. Use echo -n "text" | sha256sum for the exact string hash.

$ shred

Intermediate

Securely overwrite and optionally delete files

shred securely overwrites a file to make recovery difficult. It writes random data over the file multiple times before optionally deleting it, defeating standard file recovery tools.

shred is used for destroying sensitive data: confidential documents, old encryption keys, database dumps, and any...

Options & Flags

`-u`	Remove (unlink) file after overwriting
`-n`	Number of overwrite passes (default 3)
`-z`	Add final pass of zeros (hide shredding)
`-v`	Verbose - show progress
`-f`	Force (change permissions if needed)

Practical Examples

Example: Securely delete file

```
$ shred -vuz secret.txt
shred: secret.txt: pass 1/4 (random)...\nshred: secret.txt: pass 2/4 (random)...\nshred: secret.txt: pass 3/4 (random)...\nshred: 
```

Overwrites 3 times with random data, zeros, then deletes.

Example: Extra passes

```
$ shred -n 7 -uz classified.doc
```

7 random passes plus a zero pass, then delete.

Example: Shred multiple files

```
$ shred -vuz *.key *.pem
```

Securely destroys all key and certificate files.

Example: Wipe disk partition

```
$ sudo shred -v -n 1 /dev/sdb1
```

Overwrites an entire partition with random data.

Example: Force shred read-only file

```
$ shred -fuz protected_file.txt
```

Changes permissions if needed before shredding.

Tips & Best Practices

Warning: Not effective on SSDs and modern filesystems: shred may not work on SSDs (wear leveling), journaling filesystems (ext4), or CoW filesystems (btrfs/ZFS). Use full disk encryption instead.

Pro Tip: -z hides the shredding: -z adds a final pass of zeros, making the file look like it was zeroed rather than shredded.

Note: Full disk encryption is better: For SSD security, use full disk encryption (LUKS). Deleting the encryption key makes all data unrecoverable.

$ ufw

Beginner

Uncomplicated Firewall - easy iptables management

ufw (Uncomplicated Firewall) is a user-friendly frontend for iptables. It simplifies firewall management with straightforward commands for allowing and denying traffic by port, service name, or IP address.

ufw is the default firewall tool on Ubuntu and is designed for simplicity. It handles the ...

Options & Flags

`enable`	Enable the firewall
`disable`	Disable the firewall
`status`	Show firewall status and rules
`allow`	Allow incoming traffic
`deny`	Deny incoming traffic
`delete`	Delete a rule
`reset`	Reset all rules to defaults

Practical Examples

Example: Enable firewall

```
$ sudo ufw enable
Firewall is active and enabled on system startup
```

Activates the firewall. Make sure SSH is allowed first!

Example: Allow SSH

```
$ sudo ufw allow ssh
Rule added
```

Allows SSH connections (port 22). Always do this before enabling!

Example: Allow web traffic

```
$ sudo ufw allow 80/tcp && sudo ufw allow 443/tcp
```

Allows HTTP and HTTPS traffic.

Example: Allow from specific IP

```
$ sudo ufw allow from 192.168.1.0/24 to any port 3306
```

Allows MySQL access only from the local network.

Example: Check status

```
$ sudo ufw status numbered
[ 1] 22/tcp    ALLOW IN    Anywhere\n[ 2] 80/tcp    ALLOW IN    Anywhere
```

Shows all rules with numbers for easy deletion.

Tips & Best Practices

Warning: Allow SSH before enabling: Always run sudo ufw allow ssh BEFORE sudo ufw enable, or you will lock yourself out of remote servers.

Pro Tip: Application profiles: ufw app list shows available profiles. sudo ufw allow "Nginx Full" allows both HTTP and HTTPS for Nginx.

Note: Logging: sudo ufw logging on enables firewall logging. Check logs in /var/log/ufw.log.

Ready for more? Explore 200+ professional IT eBooks

Go deeper with comprehensive guides, hands-on projects, and real-world examples

dargslan.com/books