# 🛠️ Comandos Úteis - Blueprint Blog

Este documento contém comandos úteis para desenvolvimento, manutenção e troubleshooting do projeto.

🌐 **Domínio:** https://blueprintblog.tech

## 📋 Índice



---

## 🔄 Git & Versionamento Avançado

### 📋 Status e Histórico

```
# Status detalhado
git status -s                           # Status curto
git status --porcelain                  # Status para scripts
git status --ignored                    # Incluir arquivos ignorados

# Histórico avançado
git log --oneline --graph --all --decorate
git log --stat                          # Com estatísticas
git log --patch                         # Com diff completo
git log --since="2 weeks ago"           # Por período
git log --author="nome"                 # Por autor
git log --grep="fix"                    # Por mensagem
git log -S "função"                     # Por código adicionado/removido
git log --follow arquivo.txt            # Seguir renomeações

# Visualizar mudanças
git show HEAD                           # Último commit
git show HEAD~2                         # 2 commits atrás
git show --name-only HEAD               # Apenas nomes dos arquivos
git show --stat HEAD                    # Com estatísticas
```

## 🌿 Branches Avançadas

```
# Criar e trocar branches
git checkout -b feature/nova-funcionalidade
git switch -c feature/nova-funcionalidade  # Comando moderno

# Listar branches
git branch -a                              # Todas (local + remote)
git branch -r                              # Apenas remotas
git branch -v                              # Com último commit
git branch --merged                        # Branches já mergeadas
git branch --no-merged                     # Branches não mergeadas

# Deletar branches
git branch -d feature-branch               # Delete seguro
git branch -D feature-branch               # Force delete
git push origin --delete feature-branch # Delete remota

# Renomear branch
git branch -m old-name new-name
git branch -M old-name new-name            # Force rename
```

## 🔄 Merge e Rebase

```
# Merge strategies
git merge feature-branch                   # Merge normal
git merge --no-ff feature-branch           # Sempre criar merge commit
git merge --squash feature-branch          # Squash todos os commits

# Rebase interativo
git rebase -i HEAD~3                       # Últimos 3 commits
git rebase -i main                         # Desde main
git rebase --continue                      # Continuar após resolver conflitos
git rebase --abort                        # Cancelar rebase

# Cherry-pick
git cherry-pick abc123                     # Aplicar commit específico
git cherry-pick abc123..def456             # Range de commits
git cherry-pick --no-commit abc123         # Sem criar commit
```

## 🔍 Busca e Investigação

```
# Buscar no código
git grep "função"                          # Buscar no working tree
git grep "função" HEAD~5                   # Buscar em commit específico
git grep -n "função"                       # Com números de linha
git grep -i "função"                       # Case insensitive
```

```
# Blame e histórico de arquivo
git blame arquivo.txt                    # Quem modificou cada linha
git blame -L 10,20 arquivo.txt           # Linhas específicas
git log -p arquivo.txt                   # Histórico com patches
git log --follow arquivo.txt             # Seguir renomeações

# Bisect para encontrar bugs
git bisect start
git bisect bad HEAD                      # Commit atual tem bug
git bisect good v1.0.0                   # Versão boa conhecida
git bisect run npm test                  # Automatizar com testes
git bisect reset                         # Finalizar bisect
```

## 🏷️ Tags e Releases

```
# Criar tags
git tag v1.0.0                           # Tag simples
git tag -a v1.0.0 -m "Release 1.0.0"     # Tag anotada
git tag -a v1.0.0 abc123 -m "Tag commit específico"

# Listar e gerenciar tags
git tag                                  # Listar todas
git tag -l "v1.*"                        # Filtrar tags
git show v1.0.0                          # Ver detalhes da tag
git tag -d v1.0.0                        # Deletar tag local
git push origin --delete v1.0.0          # Deletar tag remota

# Push tags
git push origin v1.0.0                   # Tag específica
git push origin --tags                   # Todas as tags
```

## 🔧 Stash Avançado

```
# Stash básico
git stash                                # Stash mudanças
git stash push -m "WIP: feature X"       # Com mensagem
git stash push -- arquivo.txt            # Arquivo específico
git stash push --include-untracked       # Incluir arquivos não rastreados

# Gerenciar stashes
git stash list                           # Listar stashes
git stash show stash@{0}                 # Ver mudanças
git stash show -p stash@{0}              # Ver patch completo
git stash apply stash@{0}                # Aplicar sem remover
git stash pop stash@{0}                  # Aplicar e remover
git stash drop stash@{0}                 # Remover stash
git stash clear                          # Limpar todos
```

```
# Stash avançado
git stash branch nova-branch stash@{0}  # Criar branch do stash
```

### 🔄 Reset e Restore

```
# Reset (cuidado!)
git reset --soft HEAD~1                 # Desfaz commit, mantém staged
git reset --mixed HEAD~1                # Desfaz commit e staging
git reset --hard HEAD~1                 # APAGA TUDO! Cuidado!

# Restore (comando moderno)
git restore arquivo.txt                 # Restaurar working tree
git restore --staged arquivo.txt        # Unstage arquivo
git restore --source=HEAD~1 arquivo.txt # Restaurar de commit específico

# Reflog (histórico de referências)
git reflog                              # Ver histórico de HEAD
git reflog show main                    # Histórico de branch
git reset --hard HEAD@{2}              # Voltar usando reflog
```

### 🌐 Remote e Colaboração

```
# Gerenciar remotes
git remote -v                           # Listar remotes
git remote add upstream https://...      # Adicionar remote
git remote set-url origin https://...   # Alterar URL
git remote remove upstream              # Remover remote

# Fetch e pull avançado
git fetch --all                         # Fetch de todos os remotes
git fetch --prune                       # Remove refs deletadas
git pull --rebase                       # Pull com rebase
git pull --ff-only                      # Apenas fast-forward

# Push avançado
git push -u origin feature-branch       # Set upstream
git push --force-with-lease             # Force push seguro
git push origin :feature-branch         # Delete remote branch
```

---

## 📦 Node.js & Frontend

### Gerenciamento de dependências

```
# Instalar dependências
npm install
```

```
# Limpar cache
npm cache clean --force

# Verificar scripts disponíveis
npm run

# Remover node_modules e reinstalar
Remove-Item -Recurse -Force node_modules  # PowerShell
rm -rf node_modules                        # Bash
npm install
```

### Build e desenvolvimento

```
# Servidor de desenvolvimento
npm run dev

# Build de produção
npm run build

# Preview do build
npm run preview

# Verificar bundle size
npm run build -- --analyze
```

### Verificar estrutura de arquivos

```
# Contar arquivos por extensão
Get-ChildItem -Path src -Recurse -Include *.js,*.jsx | Measure-Object |
Select-Object Count
Get-ChildItem -Path src -Recurse -Include *.ts,*.tsx | Measure-Object |
Select-Object Count

# Listar arquivos específicos
Get-ChildItem -Path src -Recurse -Include *.js,*.jsx | Select-Object
Name,Directory
```

---

## 🦀 Rust & Backend

### Instalação do Rust

```
# Windows (PowerShell)
Invoke-WebRequest -Uri "https://win.rustup.rs/x86_64" -OutFile "rustup-
init.exe"
```

```
.\rustup-init.exe --default-toolchain stable --profile default --target
x86_64-pc-windows-msvc -y

# Linux/WSL
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
source ~/.cargo/env
```

### Comandos Cargo

```
# Verificar versão
cargo --version
rustc --version

# Compilar projeto
cargo build

# Executar projeto
cargo run

# Executar com logs
RUST_LOG=debug cargo run

# Verificar código
cargo check

# Executar testes
cargo test

# Atualizar dependências
cargo update
```

---

## 🔍 TypeScript

### Verificação de tipos

```
# Verificar erros sem compilar
npx tsc --noEmit

# Verificar arquivo específico
npx tsc --noEmit src/components/Component.tsx

# Gerar arquivos de declaração
npx tsc --declaration --emitDeclarationOnly
```

### Configuração

```
# Verificar configuração
npx tsc --showConfig

# Inicializar tsconfig.json
npx tsc --init
```

---

## 🐳 Docker & Docker Compose

### 📋 Comandos básicos Docker

```
# Verificar versão e informações
docker --version
docker info
docker system info

# Listar containers
docker ps                    # Containers rodando
docker ps -a                 # Todos os containers
docker ps -q                 # Apenas IDs
docker ps --format "table {{.Names}}\t{{.Status}}\t{{.Ports}}"

# Listar imagens
docker images
docker images -a             # Incluir intermediárias
docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}"
```

### 🏗️ Build e Gestão de Imagens

```
# Construir imagens
docker build -t blueprint-backend ./backend
docker build -t blueprint-frontend ./frontend
docker build --no-cache -t app:latest .
docker build --target production -t app:prod .

# Construir com argumentos
docker build --build-arg NODE_ENV=production -t app .

# Remover imagens
docker rmi image_name
docker rmi $(docker images -q)           # Todas as imagens
docker image prune                       # Imagens não utilizadas
docker image prune -a                    # Todas as imagens órfãs
```

### 🚀 Executar Containers

```
# Executar containers
docker run -p 3001:3001 blueprint-backend
docker run -d --name backend -p 3001:3001 blueprint-backend
docker run -it --rm alpine sh           # Interativo e remove ao sair
docker run -e NODE_ENV=production app   # Com variáveis de ambiente

# Executar em background
docker run -d --name nginx -p 80:80 nginx

# Executar com volumes
docker run -v $(pwd):/app -w /app node npm install
docker run -v backend_logs:/app/logs backend
```

## 🔧 Gerenciar Containers

```
# Parar containers
docker stop container_name
docker stop $(docker ps -q)             # Parar todos

# Remover containers
docker rm container_name
docker rm $(docker ps -aq)              # Remover todos
docker container prune                  # Remover containers parados

# Reiniciar containers
docker restart container_name

# Logs de containers
docker logs container_name
docker logs -f container_name            # Follow logs
docker logs --tail 50 container_name    # Últimas 50 linhas
docker logs --since="2h" container_name # Últimas 2 horas
```

## 🔍 Inspecionar e Debug

```
# Inspecionar containers/imagens
docker inspect container_name
docker inspect image_name

# Executar comandos em containers rodando
docker exec -it container_name bash
docker exec -it container_name sh
docker exec container_name ls -la /app

# Copiar arquivos
docker cp file.txt container_name:/app/
docker cp container_name:/app/logs ./logs
```

```
# Estatísticas de uso
docker stats
docker stats container_name
```

## 🧹 Limpeza e Manutenção

```
# Limpeza geral
docker system prune                    # Remove dados não utilizados
docker system prune -a                 # Remove tudo não utilizado
docker system prune --volumes          # Inclui volumes

# Limpeza específica
docker container prune                 # Containers parados
docker image prune                     # Imagens órfãs
docker volume prune                    # Volumes não utilizados
docker network prune                   # Networks não utilizadas

# Ver uso de espaço
docker system df
docker images --format "table {{.Repository}}\t{{.Tag}}\t{{.Size}}"
```

## 📦 Docker Compose - Comandos Essenciais

```
# Verificar versão
docker-compose --version

# Subir serviços
docker-compose up                      # Foreground
docker-compose up -d                   # Background (detached)
docker-compose up backend              # Serviço específico
docker-compose up --build              # Rebuild antes de subir
docker-compose up --force-recreate     # Recriar containers

# Parar serviços
docker-compose down                     # Para e remove containers
docker-compose down -v                 # Inclui volumes
docker-compose stop                    # Apenas para (não remove)
docker-compose stop backend            # Serviço específico
```

## 🔧 Docker Compose - Gestão de Serviços

```
# Logs
docker-compose logs                    # Todos os serviços
docker-compose logs -f                 # Follow logs
docker-compose logs backend            # Serviço específico
docker-compose logs --tail=50 backend  # Últimas 50 linhas
```

```
# Executar comandos
docker-compose exec backend bash        # Shell no container
docker-compose exec backend cargo build # Comando específico
docker-compose run --rm backend cargo test # Executar e remover

# Status e informações
docker-compose ps                       # Status dos serviços
docker-compose top                      # Processos rodando
docker-compose config                   # Validar configuração
docker-compose config --services        # Listar serviços
```

## 🏗️ Docker Compose - Build e Deploy

```
# Build
docker-compose build                    # Build todos os serviços
docker-compose build backend            # Build serviço específico
docker-compose build --no-cache         # Build sem cache
docker-compose build --pull             # Pull imagens base antes

# Restart e recreate
docker-compose restart                  # Restart todos
docker-compose restart backend          # Restart específico
docker-compose up -d --force-recreate   # Recriar containers

# Scale serviços
docker-compose up -d --scale backend=3  # 3 instâncias do backend
```

## 📁 Docker Compose - Arquivos Múltiplos

```
# Usar arquivos específicos
docker-compose -f docker-compose.yml up
docker-compose -f docker-compose-dev.yml up
docker-compose -f docker-compose-prod.yml up

# Combinar arquivos
docker-compose -f docker-compose.yml -f docker-compose.prod.yml up

# Especificar projeto
docker-compose -p blueprint-blog up
docker-compose -p blueprint-blog-dev up
```

## 🌐 Docker Compose - Redes e Volumes

```
# Gerenciar volumes
docker-compose down -v                  # Remove volumes
```

```
docker volume ls                          # Listar volumes
docker volume inspect volume_name         # Inspecionar volume
docker volume rm volume_name              # Remover volume

# Gerenciar redes
docker network ls                         # Listar redes
docker network inspect network_name       # Inspecionar rede
docker network rm network_name            # Remover rede
```

## 🔍 Docker Compose - Debug e Troubleshooting

```
# Validar configuração
docker-compose config                     # Mostrar configuração final
docker-compose config -q                 # Validar silenciosamente

# Debug de serviços
docker-compose ps                         # Status dos containers
docker-compose logs --tail=100 -f        # Logs em tempo real
docker-compose exec backend env           # Ver variáveis de ambiente

# Verificar conectividade
docker-compose exec frontend ping backend
docker-compose exec backend curl http://frontend:3000
```

## 🚀 Comandos Específicos do Blueprint Blog

```
# Desenvolvimento
docker-compose -f docker-compose-dev.yml up -d
docker-compose -f docker-compose-dev.yml logs -f backend
docker-compose -f docker-compose-dev.yml exec backend cargo check

# Produção
docker-compose -f docker-compose-prod.yml up -d
docker-compose -f docker-compose-prod.yml logs -f
docker-compose -f docker-compose-prod.yml exec backend curl
http://localhost:3001/health

# Build específico
docker build -t blueprint-backend:latest ./backend
docker build -t blueprint-frontend:latest ./frontend

# Logs específicos
docker-compose logs -f nginx backend frontend
docker-compose logs --since="1h" backend | grep ERROR
```

## 🔧 Comandos de Manutenção do Projeto

```
# Backup de volumes
docker run --rm -v blueprint_backend_logs:/data -v $(pwd):/backup alpine
tar czf /backup/logs-backup.tar.gz /data

# Restore de volumes
docker run --rm -v blueprint_backend_logs:/data -v $(pwd):/backup alpine
tar xzf /backup/logs-backup.tar.gz -C /

# Health checks
docker-compose exec backend curl -f http://localhost:3001/health
docker-compose exec nginx curl -f http://localhost/health

# Monitoramento
docker stats $(docker-compose ps -q)
docker-compose top
```

### 🐛 Troubleshooting Docker

```
# Verificar se Docker está rodando
docker version
systemctl status docker              # Linux
Get-Service docker                   # Windows

# Problemas de permissão (Linux)
sudo usermod -aG docker $USER
newgrp docker

# Problemas de espaço
docker system df                     # Ver uso de espaço
docker system prune -a               # Limpar tudo

# Problemas de rede
docker network ls
docker network inspect bridge

# Verificar logs do Docker daemon
journalctl -u docker.service         # Linux
Get-EventLog -LogName Application -Source Docker # Windows

# Reset completo (cuidado!)
docker system prune -a --volumes
docker builder prune -a
```

---

## 🌐 Nginx - Servidor Web & Proxy

### 📋 Comandos básicos Nginx

```
# Verificar versão e configuração
nginx -v
nginx -V                              # Versão com módulos compilados
nginx -t                              # Testar configuração
nginx -T                              # Testar e mostrar configuração

# Controlar serviço
sudo systemctl start nginx           # Iniciar
sudo systemctl stop nginx            # Parar
sudo systemctl restart nginx         # Reiniciar
sudo systemctl reload nginx          # Recarregar configuração
sudo systemctl status nginx          # Status do serviço
sudo systemctl enable nginx          # Habilitar no boot

# Comandos diretos
sudo nginx                           # Iniciar
sudo nginx -s reload                 # Recarregar
sudo nginx -s stop                   # Parar
sudo nginx -s quit                   # Parar gracefully
sudo nginx -s reopen                 # Reabrir logs
```

## 🔧 Configuração e Arquivos

```
# Localização dos arquivos (Ubuntu/Debian)
/etc/nginx/nginx.conf                # Configuração principal
/etc/nginx/sites-available/          # Sites disponíveis
/etc/nginx/sites-enabled/            # Sites habilitados
/etc/nginx/conf.d/                   # Configurações adicionais
/var/log/nginx/                      # Logs
/var/www/html/                       # Diretório web padrão

# Localização dos arquivos (CentOS/RHEL)
/etc/nginx/nginx.conf                # Configuração principal
/etc/nginx/conf.d/                   # Configurações de sites
/var/log/nginx/                      # Logs
/usr/share/nginx/html/               # Diretório web padrão

# Editar configurações
sudo nano /etc/nginx/nginx.conf
sudo nano /etc/nginx/sites-available/default
sudo nano /etc/nginx/sites-available/blueprint-blog

# Habilitar/desabilitar sites
sudo ln -s /etc/nginx/sites-available/blueprint-blog /etc/nginx/sites-
enabled/
sudo rm /etc/nginx/sites-enabled/blueprint-blog
sudo a2ensite blueprint-blog         # Habilitar site
sudo a2dissite blueprint-blog        # Desabilitar site
```

## 📊 Logs e Monitoramento

```
# Ver logs em tempo real
sudo tail -f /var/log/nginx/access.log
sudo tail -f /var/log/nginx/error.log
sudo tail -f /var/log/nginx/access.log | grep "POST"

# Logs com filtros
sudo grep "404" /var/log/nginx/access.log
sudo grep "error" /var/log/nginx/error.log
sudo grep "$(date '+%d/%b/%Y')" /var/log/nginx/access.log

# Analisar logs
sudo awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr
sudo awk '{print $7}' /var/log/nginx/access.log | sort | uniq -c | sort -nr

# Rotação de logs
sudo logrotate -f /etc/logrotate.d/nginx
sudo nginx -s reopen                # Reabrir logs após rotação
```

## 🔍 Debug e Troubleshooting

```
# Verificar sintaxe da configuração
sudo nginx -t
sudo nginx -T | grep -A 10 -B 10 "server_name"

# Verificar processos
ps aux | grep nginx
sudo netstat -tlnp | grep nginx
sudo ss -tlnp | grep nginx

# Verificar portas em uso
sudo netstat -tlnp | grep :80
sudo netstat -tlnp | grep :443
sudo lsof -i :80
sudo lsof -i :443

# Verificar configuração ativa
sudo nginx -T | grep -E "(server_name|listen|root|proxy_pass)"

# Debug de SSL
openssl s_client -connect blueprintblog.tech:443 -servername
blueprintblog.tech
sudo nginx -T | grep -A 5 -B 5 ssl
```

## 🚀 Configurações Específicas do Blueprint Blog

```

```

```
# Configuração para desenvolvimento
server {
    listen 80;
    server_name localhost;

    # Frontend (React/Next.js)
    location / {
        proxy_pass http://frontend:3000;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
        proxy_cache_bypass $http_upgrade;
    }

    # Backend API (Rust)
    location /api/ {
        proxy_pass http://backend:3001/api/;
        proxy_http_version 1.1;
        proxy_set_header Host $host;
        proxy_set_header X-Real-IP $remote_addr;
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
        proxy_set_header X-Forwarded-Proto $scheme;
    }
}

# Configuração para produção
server {
    listen 80;
    listen 443 ssl http2;
    server_name blueprintblog.tech www.blueprintblog.tech;

    # SSL Configuration
    ssl_certificate /etc/letsencrypt/live/blueprintblog.tech/fullchain.pem;
    ssl_certificate_key
/etc/letsencrypt/live/blueprintblog.tech/privkey.pem;

    # Security headers
    add_header X-Frame-Options "SAMEORIGIN" always;
    add_header X-XSS-Protection "1; mode=block" always;
    add_header X-Content-Type-Options "nosniff" always;
    add_header Referrer-Policy "no-referrer-when-downgrade" always;
    add_header Content-Security-Policy "default-src 'self' http: https:
data: blob: 'unsafe-inline'" always;

    # Gzip compression
    gzip on;
    gzip_vary on;
    gzip_min_length 1024;
    gzip_types text/plain text/css text/xml text/javascript
```

```
application/javascript application/xml+rss application/json;
}
```

### 🔧 Comandos de Manutenção Nginx

```
# Backup de configuração
sudo cp /etc/nginx/nginx.conf /etc/nginx/nginx.conf.backup
sudo tar -czf nginx-config-backup-$(date +%Y%m%d).tar.gz /etc/nginx/

# Restore de configuração
sudo cp /etc/nginx/nginx.conf.backup /etc/nginx/nginx.conf
sudo nginx -t && sudo systemctl reload nginx

# Verificar uso de recursos
sudo nginx -T | grep worker_processes
sudo nginx -T | grep worker_connections
ps aux | grep nginx | awk '{sum+=$6} END {print "Total Memory: " sum/1024 "
MB"}'

# Limpar cache (se configurado)
sudo rm -rf /var/cache/nginx/*
sudo systemctl reload nginx
```

---

## 🔧 Scripts & Automação

### 🔧 Bash Scripts Avançados

```
#!/bin/bash
# Template de script robusto

set -euo pipefail  # Exit on error, undefined vars, pipe failures
IFS=$'\n\t'        # Secure Internal Field Separator

# Variáveis globais
readonly SCRIPT_DIR="$(cd "$(dirname "${BASH_SOURCE[0]}")" && pwd)"
readonly SCRIPT_NAME="$(basename "$0")"
readonly LOG_FILE="/var/log/${SCRIPT_NAME%.sh}.log"

# Funções utilitárias
log() {
    echo "[$(date +'%Y-%m-%d %H:%M:%S')] $*" | tee -a "$LOG_FILE"
}

error() {
    log "ERROR: $*" >&2
    exit 1
}
```

```
cleanup() {
    log "Cleaning up..."
    # Cleanup code here
}

# Trap para cleanup
trap cleanup EXIT

# Verificar dependências
command -v docker >/dev/null 2>&1 || error "Docker not found"
command -v git >/dev/null 2>&1 || error "Git not found"

# Exemplo de função
deploy_blueprint() {
    local environment="${1:-dev}"
    log "Deploying Blueprint Blog to $environment"

    cd "$SCRIPT_DIR" || error "Cannot change to script directory"

    # Backup antes do deploy
    if [[ "$environment" == "prod" ]]; then
        backup_blueprint || error "Backup failed"
    fi

    # Deploy
    docker-compose -f "docker-compose-${environment}.yml" up -d || error
"Deploy failed"

    # Health check
    sleep 30
    curl -f "http://localhost/health" || error "Health check failed"

    log "Deploy completed successfully"
}

# Função de backup
backup_blueprint() {
    local backup_dir="/backup/blueprint-$(date +%Y%m%d_%H%M%S)"
    mkdir -p "$backup_dir"

    # Backup código
    tar -czf "$backup_dir/code.tar.gz" . --exclude=node_modules --
exclude=target

    # Backup volumes Docker
    docker run --rm -v blueprint_logs:/data -v "$backup_dir":/backup alpine
\
        tar czf /backup/logs.tar.gz /data

    log "Backup created at $backup_dir"
}

# Main function
main() {
```

```
    case "${1:-}" in
        deploy)
            deploy_blueprint "${2:-dev}"
            ;;
        backup)
            backup_blueprint
            ;;
        *)
            echo "Usage: $0 {deploy|backup} [environment]"
            exit 1
            ;;
    esac
}

# Execute main se script for executado diretamente
if [[ "${BASH_SOURCE[0]}" == "${0}" ]]; then
    main "$@"
fi
```

### 🐍 Python Scripts para Automação

```
#!/usr/bin/env python3
"""
Blueprint Blog Automation Script
Gerencia deploy, backup e monitoramento
"""

import os
import sys
import json
import logging
import subprocess
import argparse
from datetime import datetime
from pathlib import Path
from typing import Dict, List, Optional

# Configurar logging
logging.basicConfig(
    level=logging.INFO,
    format='%(asctime)s - %(levelname)s - %(message)s',
    handlers=[
        logging.FileHandler('/var/log/blueprint-automation.log'),
        logging.StreamHandler()
    ]
)
logger = logging.getLogger(__name__)

class BlueprintManager:
    """Gerenciador do Blueprint Blog"""
```

```
    def __init__(self, project_dir: str = "/opt/blueprint-blog"):
        self.project_dir = Path(project_dir)
        self.backup_dir = Path("/backup/blueprint")
        self.backup_dir.mkdir(parents=True, exist_ok=True)

    def run_command(self, cmd: List[str], cwd: Optional[Path] = None) ->
subprocess.CompletedProcess:
        """Executa comando com logging"""
        logger.info(f"Running: {' '.join(cmd)}")
        try:
            result = subprocess.run(
                cmd,
                cwd=cwd or self.project_dir,
                capture_output=True,
                text=True,
                check=True
            )
            logger.info(f"Command succeeded: {result.stdout}")
            return result
        except subprocess.CalledProcessError as e:
            logger.error(f"Command failed: {e.stderr}")
            raise

    def health_check(self, url: str = "http://localhost/health") -> bool:
        """Verifica saúde da aplicação"""
        try:
            result = self.run_command(["curl", "-f", url])
            return result.returncode == 0
        except subprocess.CalledProcessError:
            return False

    def backup(self) -> str:
        """Cria backup completo"""
        timestamp = datetime.now().strftime("%Y%m%d_%H%M%S")
        backup_path = self.backup_dir / f"blueprint_{timestamp}"
        backup_path.mkdir()

        # Backup código
        self.run_command([
            "tar", "-czf", str(backup_path / "code.tar.gz"),
            ".", "--exclude=node_modules", "--exclude=target"
        ])

        # Backup volumes Docker
        self.run_command([
            "docker", "run", "--rm",
            "-v", "blueprint_logs:/data",
            "-v", f"{backup_path}:/backup",
            "alpine", "tar", "czf", "/backup/logs.tar.gz", "/data"
        ])

        logger.info(f"Backup created at {backup_path}")
        return str(backup_path)
```

```
    def deploy(self, environment: str = "dev") -> None:
        """Deploy da aplicação"""
        logger.info(f"Deploying to {environment}")

        # Backup em produção
        if environment == "prod":
            self.backup()

        # Pull latest code
        self.run_command(["git", "pull", "origin", "main"])

        # Deploy
        compose_file = f"docker-compose-{environment}.yml"
        self.run_command(["docker-compose", "-f", compose_file, "up", "-d",
"--build"])

        # Health check
        import time
        time.sleep(30)
        if not self.health_check():
            raise Exception("Health check failed after deploy")

        logger.info("Deploy completed successfully")

    def monitor(self) -> Dict:
        """Monitora status da aplicação"""
        status = {
            "timestamp": datetime.now().isoformat(),
            "health": self.health_check(),
            "containers": [],
            "disk_usage": {},
            "memory_usage": {}
        }

        # Status containers
        try:
            result = self.run_command(["docker-compose", "ps", "--format",
"json"])
            status["containers"] = json.loads(result.stdout)
        except:
            pass

        # Uso de disco
        try:
            result = self.run_command(["df", "-h", "/"])
            status["disk_usage"] = result.stdout
        except:
            pass

        return status

def main():
    parser = argparse.ArgumentParser(description="Blueprint Blog Manager")
    parser.add_argument("action", choices=["deploy", "backup", "monitor",
```

```
"health"])
    parser.add_argument("--env", default="dev", help="Environment
(dev/prod)")
    parser.add_argument("--project-dir", default="/opt/blueprint-blog",
help="Project directory")

    args = parser.parse_args()

    manager = BlueprintManager(args.project_dir)

    try:
        if args.action == "deploy":
            manager.deploy(args.env)
        elif args.action == "backup":
            backup_path = manager.backup()
            print(f"Backup created: {backup_path}")
        elif args.action == "monitor":
            status = manager.monitor()
            print(json.dumps(status, indent=2))
        elif args.action == "health":
            healthy = manager.health_check()
            print("Healthy" if healthy else "Unhealthy")
            sys.exit(0 if healthy else 1)
    except Exception as e:
        logger.error(f"Action failed: {e}")
        sys.exit(1)

if __name__ == "__main__":
    main()
```

## 🔄 Systemd Services

```
# /etc/systemd/system/blueprint-monitor.service
[Unit]
Description=Blueprint Blog Monitor
After=docker.service
Requires=docker.service

[Service]
Type=simple
User=blueprint
Group=blueprint
WorkingDirectory=/opt/blueprint-blog
ExecStart=/usr/local/bin/blueprint-manager monitor
Restart=always
RestartSec=60
StandardOutput=journal
StandardError=journal

[Install]
WantedBy=multi-user.target
```

```
# Comandos para gerenciar
sudo systemctl daemon-reload
sudo systemctl enable blueprint-monitor
sudo systemctl start blueprint-monitor
sudo systemctl status blueprint-monitor
```

## ⏰ Cron Jobs Avançados

```
# /etc/cron.d/blueprint-blog
# Backup diário às 2h
0 2 * * * blueprint /usr/local/bin/blueprint-manager backup >>
/var/log/blueprint-cron.log 2>&1

# Health check a cada 5 minutos
*/5 * * * * blueprint /usr/local/bin/blueprint-manager health || echo
"Health check failed $(date)" >> /var/log/blueprint-alerts.log

# Limpeza de logs semanalmente
0 3 * * 0 root find /var/log -name "*.log" -mtime +7 -delete

# Renovação SSL mensal
0 4 1 * * root /usr/bin/certbot renew --quiet

# Limpeza Docker semanal
0 5 * * 0 blueprint docker system prune -f >> /var/log/docker-cleanup.log
2>&1

# Monitoramento de espaço em disco
0 */6 * * * root df -h | awk '$5 > 80 {print "Disk usage warning: " $0}' |
mail -s "Disk Space Alert" admin@blueprintblog.tech
```

## 🔔 Scripts de Alertas

```
#!/bin/bash
# /opt/scripts/alert-manager.sh

send_alert() {
    local message="$1"
    local severity="${2:-INFO}"
    local timestamp=$(date '+%Y-%m-%d %H:%M:%S')

    # Log local
    echo "[$timestamp] [$severity] $message" >> /var/log/blueprint-
alerts.log

    # Slack webhook (opcional)
    if [[ -n "$SLACK_WEBHOOK" ]]; then
        curl -X POST -H 'Content-type: application/json' \
```

```
            --data "{\"text\":\"[$severity] Blueprint Blog: $message\"}" \
            "$SLACK_WEBHOOK"
    fi

    # Email (opcional)
    if command -v mail >/dev/null 2>&1; then
        echo "$message" | mail -s "[$severity] Blueprint Blog Alert"
admin@blueprintblog.tech
    fi
}

check_service_health() {
    local service="$1"
    local url="$2"

    if ! curl -f "$url" >/dev/null 2>&1; then
        send_alert "Service $service is down (URL: $url)" "CRITICAL"
        return 1
    fi
    return 0
}

check_disk_space() {
    local threshold="${1:-85}"
    local usage=$(df / | awk 'NR==2 {print $5}' | sed 's/%//')

    if [[ $usage -gt $threshold ]]; then
        send_alert "Disk usage is ${usage}% (threshold: ${threshold}%)"
"WARNING"
        return 1
    fi
    return 0
}

check_memory_usage() {
    local threshold="${1:-90}"
    local usage=$(free | awk 'NR==2{printf "%.0f", $3*100/$2}')

    if [[ $usage -gt $threshold ]]; then
        send_alert "Memory usage is ${usage}% (threshold: ${threshold}%)"
"WARNING"
        return 1
    fi
    return 0
}

# Verificações principais
main() {
    check_service_health "Frontend" "http://localhost/"
    check_service_health "Backend" "http://localhost/api/health"
    check_disk_space 85
    check_memory_usage 90

    # Verificar containers Docker
```

```
    if ! docker-compose ps | grep -q "Up"; then
        send_alert "Some Docker containers are not running" "CRITICAL"
    fi
}

main "$@"
```

---

## 🐧 Linux - Sistema e Administração

### 📋 Comandos básicos do Sistema

```
# Informações do sistema
uname -a                              # Informações do kernel
lsb_release -a                        # Informações da distribuição
hostnamectl                           # Informações do host
uptime                                # Tempo de atividade
whoami                                # Usuário atual
id                                    # ID do usuário e grupos
w                                     # Usuários logados
last                                  # Últimos logins

# Informações de hardware
lscpu                                 # Informações da CPU
free -h                               # Uso de memória
df -h                                 # Uso de disco
lsblk                                 # Dispositivos de bloco
lsusb                                 # Dispositivos USB
lspci                                 # Dispositivos PCI
```

### 📁 Navegação e Arquivos

```
# Navegação
pwd                                   # Diretório atual
ls -la                                # Listar arquivos detalhado
ls -lah                               # Com tamanhos legíveis
tree                                  # Estrutura em árvore
find /path -name "*.log"              # Buscar arquivos
locate filename                      # Localizar arquivos (updatedb)

# Manipulação de arquivos
cp file1 file2                        # Copiar arquivo
cp -r dir1 dir2                       # Copiar diretório
mv file1 file2                        # Mover/renomear
rm file                               # Remover arquivo
rm -rf directory                      # Remover diretório
mkdir -p path/to/dir                  # Criar diretórios
chmod 755 file                        # Alterar permissões
chown user:group file                 # Alterar proprietário
```

## 🔍 Busca e Filtros

```
# Grep - busca em arquivos
grep "error" /var/log/nginx/error.log
grep -r "function" /path/to/code/
grep -i "warning" *.log              # Case insensitive
grep -v "debug" app.log               # Inverter busca
grep -n "TODO" *.js                  # Mostrar números de linha
grep -A 5 -B 5 "error" app.log       # Contexto (5 linhas antes/depois)

# Find - buscar arquivos
find . -name "*.js" -type f
find . -size +100M                   # Arquivos maiores que 100MB
find . -mtime -7                     # Modificados nos últimos 7 dias
find . -name "*.log" -exec rm {} \;  # Executar comando nos resultados

# Awk e Sed
awk '{print $1}' access.log          # Imprimir primeira coluna
sed 's/old/new/g' file.txt           # Substituir texto
sed -i 's/old/new/g' file.txt        # Substituir in-place
```

## 🔧 Processos e Serviços

```
# Processos
ps aux                               # Listar todos os processos
ps aux | grep nginx                  # Filtrar processos
top                                  # Monitor de processos em tempo real
htop                                 # Monitor melhorado (se instalado)
kill PID                             # Matar processo por PID
killall nginx                       # Matar todos os processos nginx
pgrep nginx                          # Encontrar PID por nome
pkill nginx                          # Matar processo por nome

# Serviços (systemd)
sudo systemctl status nginx         # Status do serviço
sudo systemctl start nginx          # Iniciar serviço
sudo systemctl stop nginx           # Parar serviço
sudo systemctl restart nginx        # Reiniciar serviço
sudo systemctl enable nginx         # Habilitar no boot
sudo systemctl disable nginx        # Desabilitar no boot
sudo systemctl list-units --type=service # Listar serviços
```

## 🌐 Rede e Conectividade

```
# Informações de rede
ip addr show                         # Interfaces de rede
```

```
ip route show                          # Tabela de roteamento
netstat -tlnp                          # Portas abertas
ss -tlnp                               # Alternativa moderna ao netstat
lsof -i :80                            # Processos usando porta 80

# Conectividade
ping google.com                        # Testar conectividade
curl -I http://blueprintblog.tech      # Testar HTTP
wget http://example.com/file           # Baixar arquivo
nslookup blueprintblog.tech            # Resolver DNS
dig blueprintblog.tech                 # DNS lookup detalhado

# Firewall (UFW)
sudo ufw status                        # Status do firewall
sudo ufw enable                        # Habilitar firewall
sudo ufw allow 80                      # Permitir porta 80
sudo ufw allow 443                     # Permitir porta 443
sudo ufw deny 22                       # Negar porta 22
sudo ufw delete allow 80               # Remover regra
```

## 📊 Monitoramento e Performance

```
# CPU e Memória
top                                    # Monitor em tempo real
htop                                   # Monitor melhorado
vmstat 1                               # Estatísticas de VM
iostat 1                               # Estatísticas de I/O
sar -u 1 10                            # CPU usage (10 samples)

# Disco
df -h                                  # Uso de disco
du -sh /path/to/dir                    # Tamanho de diretório
du -h --max-depth=1 /var/log           # Tamanho por subdiretório
iotop                                  # Monitor de I/O por processo
lsof +D /path/to/dir                   # Arquivos abertos em diretório

# Rede
iftop                                  # Monitor de tráfego de rede
nethogs                                # Uso de rede por processo
ss -s                                  # Estatísticas de sockets
```

## 🔐 Usuários e Permissões

```
# Usuários
sudo adduser username                  # Adicionar usuário
sudo deluser username                  # Remover usuário
sudo usermod -aG sudo username         # Adicionar ao grupo sudo
su - username                          # Trocar de usuário
sudo -u username command               # Executar como outro usuário
```

```
# Grupos
groups                                  # Grupos do usuário atual
sudo addgroup groupname                 # Criar grupo
sudo usermod -aG groupname username     # Adicionar usuário ao grupo

# Permissões
chmod 755 file                          # rwxr-xr-x
chmod u+x file                          # Adicionar execução para owner
chmod g-w file                          # Remover escrita para grupo
chown user:group file                   # Alterar proprietário
chown -R user:group directory           # Recursivo
```

## 📦 Gerenciamento de Pacotes

```
# Ubuntu/Debian (APT)
sudo apt update                         # Atualizar lista de pacotes
sudo apt upgrade                        # Atualizar pacotes
sudo apt install package                # Instalar pacote
sudo apt remove package                 # Remover pacote
sudo apt purge package                  # Remover pacote e configurações
sudo apt autoremove                     # Remover dependências órfãs
sudo apt search keyword                 # Buscar pacotes
apt list --installed                    # Listar pacotes instalados

# CentOS/RHEL (YUM/DNF)
sudo yum update                         # Atualizar pacotes
sudo yum install package                # Instalar pacote
sudo yum remove package                 # Remover pacote
sudo yum search keyword                 # Buscar pacotes
yum list installed                      # Listar pacotes instalados

# Snap packages
sudo snap install package               # Instalar snap
sudo snap remove package                # Remover snap
snap list                               # Listar snaps instalados
```

## 🔧 Comandos Específicos para Blueprint Blog

```
# Instalação de dependências do projeto
sudo apt update && sudo apt install -y curl wget git build-essential

# Instalar Docker
curl -fsSL https://get.docker.com -o get-docker.sh
sudo sh get-docker.sh
sudo usermod -aG docker $USER

# Instalar Docker Compose
sudo curl -L
```

```
"https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)" -o /usr/local/bin/docker-compose
sudo chmod +x /usr/local/bin/docker-compose

# Instalar Node.js (via NodeSource)
curl -fsSL https://deb.nodesource.com/setup_20.x | sudo -E bash -
sudo apt-get install -y nodejs

# Instalar Rust
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
source ~/.cargo/env

# Instalar Nginx
sudo apt install nginx
sudo systemctl enable nginx
sudo systemctl start nginx

# Verificar instalações
docker --version
docker-compose --version
node --version
npm --version
cargo --version
nginx -v
```

## 🚀 Deploy e Produção

```
# Preparar servidor para Blueprint Blog
sudo mkdir -p /opt/blueprint-blog
sudo chown $USER:$USER /opt/blueprint-blog
cd /opt/blueprint-blog

# Clonar repositório
git clone https://github.com/user/blueprint-blog.git .

# Configurar variáveis de ambiente
cp .env.example .env
nano .env

# Build e deploy
docker-compose -f docker-compose-prod.yml build
docker-compose -f docker-compose-prod.yml up -d

# Verificar status
docker-compose -f docker-compose-prod.yml ps
docker-compose -f docker-compose-prod.yml logs -f

# Configurar SSL com Let's Encrypt
sudo apt install certbot python3-certbot-nginx
sudo certbot --nginx -d blueprintblog.tech -d www.blueprintblog.tech
```

```
# Renovação automática SSL
sudo crontab -e
# Adicionar: 0 12 * * * /usr/bin/certbot renew --quiet
```

### 🔍 Troubleshooting Linux Específico

```
# Verificar logs do sistema
sudo journalctl -u nginx -f
sudo journalctl -u docker -f
sudo journalctl --since "1 hour ago"

# Verificar espaço em disco
df -h
du -sh /var/log/*
du -sh /var/lib/docker/*

# Limpar logs antigos
sudo journalctl --vacuum-time=7d
sudo find /var/log -name "*.log" -type f -mtime +30 -delete

# Verificar conectividade de rede
ping -c 4 8.8.8.8
curl -I https://blueprintblog.tech
telnet blueprintblog.tech 80
telnet blueprintblog.tech 443

# Verificar DNS
nslookup blueprintblog.tech
dig blueprintblog.tech A
dig blueprintblog.tech AAAA

# Verificar certificados SSL
openssl s_client -connect blueprintblog.tech:443 -servername
blueprintblog.tech
curl -vI https://blueprintblog.tech

# Monitorar recursos em tempo real
watch -n 1 'free -h && echo && df -h && echo && docker stats --no-stream'
```

---

## 📁 Sistema & Arquivos

### PowerShell

```
# Navegar e listar
Get-Location
Get-ChildItem -Path . -Force
Get-ChildItem -Path backend -Recurse
```

```
# Mover arquivos
Move-Item "src\file.jsx" "src\file.tsx"

# Remover arquivos/pastas
Remove-Item -Recurse -Force node_modules
Remove-Item file.txt

# Verificar conteúdo
Get-Content package.json | Select-String -A 5 -B 1 "scripts"
type package.json
```

### Bash/WSL

```
# Navegar e listar
pwd
ls -la
find . -name "*.tsx" -type f

# Mover arquivos
mv src/file.jsx src/file.tsx

# Remover arquivos/pastas
rm -rf node_modules
rm file.txt

# Verificar conteúdo
cat package.json
grep -A 5 -B 1 "scripts" package.json
```

---

## 🔧 Troubleshooting

### Problemas comuns

#### Frontend não carrega (404)

```
# Verificar se o servidor está rodando
npm run dev

# Limpar cache e reinstalar
npm cache clean --force
rm -rf node_modules
npm install

# Verificar porta
netstat -ano | findstr :5173  # Windows
lsof -i :5173                 # Linux/Mac
```

### Erros de TypeScript

```
# Verificar erros
npx tsc --noEmit

# Limpar cache do TypeScript
rm -rf node_modules/.cache
```

### Problemas com Rust

```
# Verificar instalação
cargo --version

# Limpar build
cargo clean

# Atualizar Rust
rustup update
```

### Problemas de PATH

```
# Windows - Adicionar ao PATH
$env:PATH += ";$env:USERPROFILE\.cargo\bin"

# Verificar PATH
echo $env:PATH
```

```
# Linux/WSL - Adicionar ao PATH
export PATH="$HOME/.cargo/bin:$PATH"
echo 'export PATH="$HOME/.cargo/bin:$PATH"' >> ~/.bashrc

# Verificar PATH
echo $PATH
```

---

## 📊 Monitoramento

### Performance

```
# Verificar tamanho do bundle
npm run build
ls -lh dist/assets/
```

```
# Analisar dependências
npm ls
npm audit
```

### Logs

```
# Logs do backend (Rust)
RUST_LOG=debug cargo run

# Logs do frontend
npm run dev -- --debug
```

---

## 🎯 Comandos Específicos do Projeto

### Migração TypeScript

```
# Verificar arquivos JS/JSX restantes
Get-ChildItem -Path src -Recurse -Include *.js,*.jsx | Measure-Object

# Renomear arquivos
Move-Item "Component.jsx" "Component.tsx"
```

### Configuração do ambiente

```
# Copiar arquivo de exemplo
cp .env.example .env

# Verificar configuração
cat .env
```

### Comandos específicos do domínio blueprintblog.tech

```
# Testar conectividade do domínio
nslookup blueprintblog.tech
curl -I https://blueprintblog.tech/health

# Verificar certificado SSL
openssl s_client -connect blueprintblog.tech:443 -servername
blueprintblog.tech

# Testar APIs
curl -X GET https://blueprintblog.tech/api/health
```

```
curl -X GET https://blueprintblog.tech/api/posts

# Deploy para produção
make deploy-prod

# Verificar logs de produção
docker-compose logs -f nginx
docker-compose logs -f backend
```

---

## 🥷 Comandos Ninja Avançados

### Git Avançado

```
# Buscar em todo o histórico
git log --all --grep="TypeScript"
git log --oneline --graph --all --decorate

# Encontrar quando um arquivo foi deletado
git log --full-history -- path/to/file

# Buscar código em commits
git log -S "função específica" --source --all

# Desfazer commits (cuidado!)
git reset --soft HEAD~1  # Mantém mudanças staged
git reset --hard HEAD~1  # APAGA TUDO!

# Stash avançado
git stash push -m "WIP: feature X" -- src/
git stash list
git stash apply stash@{0}

# Rebase interativo (reescrever história)
git rebase -i HEAD~3

# Cherry-pick commits específicos
git cherry-pick abc123def

# Bisect para encontrar bugs
git bisect start
git bisect bad HEAD
git bisect good v1.0.0
```

### Node.js Ninja

```
# Analisar dependências duplicadas
npm ls --depth=0
npm dedupe
```

```
# Verificar vulnerabilidades
npm audit --audit-level high
npm audit fix --force

# Executar scripts em paralelo
npm run dev & npm run backend

# Verificar qual processo usa uma porta
netstat -ano | findstr :3000
taskkill /PID 1234 /F

# NPM com diferentes registries
npm install --registry https://registry.npmjs.org/
npm config set registry https://registry.npmjs.org/

# Limpar tudo (nuclear option)
npm cache clean --force
rm -rf node_modules package-lock.json
npm install

# Verificar tamanho de pacotes
npm install -g bundlephobia-cli
bundlephobia react

# Executar scripts sem npm
npx --no-install vite build
```

## PowerShell Ninja

```
# Buscar texto em arquivos
Select-String -Path "src\**\*.tsx" -Pattern "useState"
Get-ChildItem -Recurse -Include "*.ts","*.tsx" | Select-String "interface"

# Operações em massa
Get-ChildItem -Filter "*.jsx" -Recurse | ForEach-Object {
    Rename-Item $_.FullName ($_.Name -replace "\.jsx$", ".tsx")
}

# Monitorar mudanças em arquivos
Get-ChildItem -Path "src" -Recurse | Get-FileHash | Export-Csv baseline.csv

# Comparar diretórios
Compare-Object (Get-ChildItem src) (Get-ChildItem backup) -Property Name

# Executar comandos em paralelo
Start-Job -ScriptBlock { npm run build }
Get-Job | Receive-Job

# Variáveis de ambiente avançadas
[Environment]::SetEnvironmentVariable("NODE_ENV", "development", "User")
```

```
Get-ChildItem Env: | Where-Object Name -like "*NODE*"

# Aliases úteis
Set-Alias ll Get-ChildItem
Set-Alias grep Select-String

# Histórico de comandos
Get-History | Where-Object CommandLine -like "*npm*"
```

### Bash/WSL Ninja

```
# Find avançado
find . -name "*.tsx" -exec grep -l "useState" {} \;
find . -type f -name "*.js" -delete

# Sed para substituições em massa
find src -name "*.tsx" -exec sed -i 's/React.FC/FC/g' {} \;

# Awk para processamento de texto
npm ls --json | jq '.dependencies | keys[]'

# Xargs para operações em lote
find . -name "*.log" | xargs rm
ls *.jsx | xargs -I {} mv {} {}.backup

# Monitoramento em tempo real
watch -n 2 'npm run build'
tail -f logs/app.log | grep ERROR

# Aliases ninja
alias ll='ls -alF'
alias la='ls -A'
alias l='ls -CF'
alias ..='cd ..'
```

---

## 🌐 Redes & Conectividade

### 🔍 Diagnóstico de Rede

```
# Informações básicas de rede
ip addr show                          # Interfaces e IPs
ip route show                         # Tabela de roteamento
ip link show                          # Status das interfaces
ifconfig                              # Informações de interface (legacy)

# Conectividade básica
ping -c 4 8.8.8.8                    # Teste de conectividade
ping -c 4 blueprintblog.tech          # Teste de domínio
```

```
traceroute blueprintblog.tech        # Rastreamento de rota
mtr blueprintblog.tech               # Traceroute contínuo

# DNS
nslookup blueprintblog.tech          # Lookup básico
dig blueprintblog.tech               # Lookup detalhado
dig @8.8.8.8 blueprintblog.tech      # Usar DNS específico
dig blueprintblog.tech ANY           # Todos os registros
host blueprintblog.tech              # Lookup simples

# Portas e conexões
netstat -tlnp                        # Portas TCP abertas
ss -tlnp                             # Alternativa moderna
lsof -i :80                          # Processos na porta 80
nmap -p 80,443 blueprintblog.tech    # Scan de portas
telnet blueprintblog.tech 80         # Teste de porta
nc -zv blueprintblog.tech 80         # Netcat port check
```

## 🔧 Configuração de Rede

```
# Configurar interface (temporário)
sudo ip addr add 192.168.1.100/24 dev eth0
sudo ip link set eth0 up
sudo ip route add default via 192.168.1.1

# Configuração permanente (Ubuntu/Debian)
sudo nano /etc/netplan/01-netcfg.yaml
sudo netplan apply

# Configuração DNS
sudo nano /etc/resolv.conf
echo "nameserver 8.8.8.8" | sudo tee -a /etc/resolv.conf

# Reiniciar serviços de rede
sudo systemctl restart networking
sudo systemctl restart NetworkManager
```

## 🌐 Teste de Conectividade Web

```
# HTTP/HTTPS básico
curl -I http://blueprintblog.tech     # Headers apenas
curl -v https://blueprintblog.tech    # Verbose
curl -L https://blueprintblog.tech    # Seguir redirects
curl -o page.html https://blueprintblog.tech  # Salvar em arquivo

# Teste de APIs
curl -X GET https://blueprintblog.tech/api/health
curl -X POST -H "Content-Type: application/json" \
     -d '{"test": "data"}' \
```

```
     https://blueprintblog.tech/api/test

# Teste de performance
curl -w "@curl-format.txt" -o /dev/null -s https://blueprintblog.tech
time curl -o /dev/null -s https://blueprintblog.tech

# Wget alternativo
wget --spider https://blueprintblog.tech  # Apenas verificar
wget -O - https://blueprintblog.tech      # Output para stdout
wget -r -l 2 https://blueprintblog.tech   # Download recursivo
```

## 🔒 SSL/TLS e Certificados

```
# Verificar certificado SSL
openssl s_client -connect blueprintblog.tech:443 -servername
blueprintblog.tech
openssl s_client -connect blueprintblog.tech:443 -showcerts

# Informações do certificado
echo | openssl s_client -connect blueprintblog.tech:443 2>/dev/null | \
openssl x509 -noout -dates

# Verificar expiração
echo | openssl s_client -connect blueprintblog.tech:443 2>/dev/null | \
openssl x509 -noout -enddate

# Teste de cipher suites
nmap --script ssl-enum-ciphers -p 443 blueprintblog.tech

# Verificar configuração SSL
curl -I --tlsv1.2 https://blueprintblog.tech
curl -I --tlsv1.3 https://blueprintblog.tech
```

## 🔥 Firewall e Segurança

```
# UFW (Ubuntu Firewall)
sudo ufw status verbose             # Status detalhado
sudo ufw enable                     # Habilitar
sudo ufw default deny incoming      # Política padrão
sudo ufw default allow outgoing     # Política padrão

# Regras específicas
sudo ufw allow 22/tcp               # SSH
sudo ufw allow 80/tcp               # HTTP
sudo ufw allow 443/tcp              # HTTPS
sudo ufw allow from 192.168.1.0/24  # Subnet específica
sudo ufw deny from 192.168.1.100    # IP específico

# Regras avançadas
```

```
sudo ufw allow in on eth0 to any port 22
sudo ufw limit ssh                    # Rate limiting
sudo ufw delete allow 80              # Remover regra

# iptables (avançado)
sudo iptables -L -n                   # Listar regras
sudo iptables -A INPUT -p tcp --dport 80 -j ACCEPT
sudo iptables-save > /etc/iptables/rules.v4
```

## 📊 Monitoramento de Rede

```
# Tráfego em tempo real
iftop                                 # Por interface
nethogs                               # Por processo
nload                                 # Gráfico simples
bmon                                  # Monitor avançado

# Estatísticas de rede
ss -s                                 # Resumo de sockets
netstat -i                            # Estatísticas de interface
cat /proc/net/dev                     # Estatísticas do kernel

# Captura de pacotes
sudo tcpdump -i eth0 port 80          # Capturar porta 80
sudo tcpdump -i any host blueprintblog.tech  # Host específico
sudo tcpdump -w capture.pcap -i eth0  # Salvar em arquivo
wireshark                             # Interface gráfica

# Análise de logs de rede
sudo tail -f /var/log/syslog | grep -i network
journalctl -u NetworkManager -f
```

## 🐳 Rede Docker

```
# Redes Docker
docker network ls                     # Listar redes
docker network inspect bridge        # Inspecionar rede
docker network create --driver bridge blueprint-net

# Conectividade entre containers
docker exec -it container1 ping container2
docker exec -it frontend curl http://backend:3001/health

# Debug de rede Docker
docker run --rm --net container:backend nicolaka/netshoot
docker exec -it backend netstat -tlnp
docker logs backend | grep -i network

# Port mapping
```

```
docker run -p 8080:80 nginx          # Mapear porta
docker port container_name           # Ver mapeamentos
```

---

## 📊 Logs & Monitoramento

### 📝 Sistema de Logs Linux

```
# Journalctl (systemd)
journalctl -f                        # Follow logs
journalctl -u nginx                  # Serviço específico
journalctl -u nginx -f               # Follow serviço
journalctl --since "1 hour ago"      # Por período
journalctl --since "2023-01-01"      # Data específica
journalctl -p err                    # Por prioridade (err, warning, info)
journalctl -n 50                     # Últimas 50 linhas
journalctl --disk-usage              # Uso de espaço

# Logs tradicionais
tail -f /var/log/syslog              # Log do sistema
tail -f /var/log/auth.log            # Logs de autenticação
tail -f /var/log/nginx/access.log    # Logs do Nginx
tail -f /var/log/nginx/error.log     # Erros do Nginx

# Múltiplos logs simultaneamente
multitail /var/log/syslog /var/log/nginx/access.log
tail -f /var/log/nginx/*.log

# Rotação de logs
sudo logrotate -f /etc/logrotate.d/nginx
sudo journalctl --vacuum-time=7d     # Limpar logs antigos
```

### 🔍 Análise de Logs

```
# Grep avançado em logs
grep -i "error" /var/log/nginx/error.log
grep -E "(error|warning|critical)" /var/log/syslog
grep -v "debug" /var/log/app.log     # Excluir debug

# Análise com awk
awk '{print $1}' /var/log/nginx/access.log | sort | uniq -c | sort -nr
awk '$9 >= 400 {print $0}' /var/log/nginx/access.log  # Status 4xx/5xx

# Análise temporal
grep "$(date '+%b %d')" /var/log/syslog  # Logs de hoje
grep "$(date -d yesterday '+%b %d')" /var/log/syslog  # Ontem

# Estatísticas de logs
wc -l /var/log/nginx/access.log      # Contar linhas
```

```
du -sh /var/log/*                    # Tamanho dos logs
find /var/log -name "*.log" -mtime +7  # Logs antigos
```

### 🐳 Logs Docker

```
# Logs de containers
docker logs container_name           # Logs do container
docker logs -f container_name        # Follow logs
docker logs --tail 50 container_name # Últimas 50 linhas
docker logs --since="1h" container_name  # Última hora
docker logs --timestamps container_name  # Com timestamps

# Docker Compose logs
docker-compose logs                  # Todos os serviços
docker-compose logs -f               # Follow todos
docker-compose logs backend          # Serviço específico
docker-compose logs --tail=100 -f    # Últimas 100 linhas

# Configuração de logging
docker run --log-driver=json-file --log-opt max-size=10m nginx
docker run --log-driver=syslog nginx

# Logs centralizados
docker run -d --name logspout \
  -v /var/run/docker.sock:/var/run/docker.sock \
  gliderlabs/logspout syslog://logserver:514
```

### 📊 Monitoramento de Sistema

```
# CPU e Memória
top                                  # Monitor básico
htop                                 # Monitor melhorado
vmstat 1                             # Estatísticas de VM
iostat 1                             # I/O statistics
sar -u 1 10                          # CPU usage

# Processos
ps aux --sort=-%cpu | head -10       # Top CPU
ps aux --sort=-%mem | head -10       # Top Memory
pstree                               # Árvore de processos

# Disco
df -h                                # Uso de disco
du -sh /var/log/*                    # Tamanho por diretório
iotop                                # I/O por processo
lsof +D /var/log                     # Arquivos abertos

# Rede
iftop                                # Tráfego de rede
```

```
nethogs                                 # Uso por processo
ss -tuln                                # Sockets ativos
```

## 🔔 Alertas e Notificações

```
# Script de monitoramento básico
#!/bin/bash
# /opt/scripts/monitor.sh

# Verificar uso de CPU
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d'%' -f1)
if (( $(echo "$cpu_usage > 80" | bc -l) )); then
    echo "High CPU usage: $cpu_usage%" | mail -s "CPU Alert"
admin@blueprintblog.tech
fi

# Verificar uso de memória
mem_usage=$(free | awk 'NR==2{printf "%.0f", $3*100/$2}')
if [ $mem_usage -gt 90 ]; then
    echo "High memory usage: $mem_usage%" | mail -s "Memory Alert"
admin@blueprintblog.tech
fi

# Verificar espaço em disco
disk_usage=$(df / | awk 'NR==2{print $5}' | cut -d'%' -f1)
if [ $disk_usage -gt 85 ]; then
    echo "High disk usage: $disk_usage%" | mail -s "Disk Alert"
admin@blueprintblog.tech
fi

# Verificar serviços
if ! systemctl is-active --quiet nginx; then
    echo "Nginx is down!" | mail -s "Service Alert"
admin@blueprintblog.tech
fi

if ! docker ps | grep -q blueprint; then
    echo "Blueprint containers are down!" | mail -s "Docker Alert"
admin@blueprintblog.tech
fi
```

## 📈 Métricas e Performance

```
# Coleta de métricas
#!/bin/bash
# /opt/scripts/collect-metrics.sh

timestamp=$(date '+%Y-%m-%d %H:%M:%S')
cpu_usage=$(top -bn1 | grep "Cpu(s)" | awk '{print $2}' | cut -d'%' -f1)
```

```
mem_usage=$(free | awk 'NR==2{printf "%.0f", $3*100/$2}')
disk_usage=$(df / | awk 'NR==2{print $5}' | cut -d'%' -f1)
load_avg=$(uptime | awk -F'load average:' '{print $2}')

# Salvar métricas
echo "$timestamp,$cpu_usage,$mem_usage,$disk_usage,$load_avg" >>
/var/log/metrics.csv

# Métricas Docker
docker stats --no-stream --format "table
{{.Container}}\t{{.CPUPerc}}\t{{.MemUsage}}" >> /var/log/docker-metrics.log

# Métricas de rede
rx_bytes=$(cat /sys/class/net/eth0/statistics/rx_bytes)
tx_bytes=$(cat /sys/class/net/eth0/statistics/tx_bytes)
echo "$timestamp,$rx_bytes,$tx_bytes" >> /var/log/network-metrics.csv
```

## 🔧 Log Management Avançado

```
# Configuração de logrotate personalizada
# /etc/logrotate.d/blueprint-blog
/var/log/blueprint-blog/*.log {
    daily
    missingok
    rotate 30
    compress
    delaycompress
    notifempty
    create 644 blueprint blueprint
    postrotate
        systemctl reload nginx
        docker-compose -f /opt/blueprint-blog/docker-compose-prod.yml
restart
    endscript
}

# Centralização de logs com rsyslog
# /etc/rsyslog.d/50-blueprint.conf
$ModLoad imfile
$InputFileName /var/log/blueprint-blog/app.log
$InputFileTag blueprint-app:
$InputFileStateFile stat-blueprint-app
$InputFileSeverity info
$InputFileFacility local0
$InputRunFileMonitor

# Enviar para servidor central
*.* @@logserver.blueprintblog.tech:514

# Análise de logs com jq (para JSON logs)
```

```
tail -f /var/log/blueprint-blog/app.log | jq '.level, .message'
cat /var/log/blueprint-blog/app.log | jq 'select(.level == "error")'
```

### 🚨 Troubleshooting com Logs

```
# Diagnóstico rápido
#!/bin/bash
# /opt/scripts/quick-diagnosis.sh

echo "=== System Status ==="
uptime
free -h
df -h

echo "=== Service Status ==="
systemctl status nginx docker

echo "=== Recent Errors ==="
journalctl -p err --since "1 hour ago" --no-pager

echo "=== Docker Status ==="
docker ps
docker-compose ps

echo "=== Network Status ==="
ss -tuln | grep -E "(80|443|3001)"

echo "=== Recent Access Logs ==="
tail -20 /var/log/nginx/access.log

echo "=== Recent Error Logs ==="
tail -20 /var/log/nginx/error.log

# Verificar conectividade
curl -I http://localhost/health 2>/dev/null || echo "Health check failed"
```

---

## 📁 Arquivos & Diretórios Avançado

### 🔍 Busca e Localização Avançada

```
# Find com múltiplos critérios
find /path -name "*.log" -type f -size +10M -mtime -7
find . -name "*.js" -o -name "*.ts" -o -name "*.jsx" -o -name "*.tsx"
find . -type f -name "*.txt" -exec grep -l "pattern" {} \;
find . -type f -name "*.log" -exec rm {} \;
find . -type d -name "node_modules" -exec rm -rf {} +

# Find com ações avançadas
```

```
find . -name "*.tmp" -delete                    # Deletar arquivos
find . -type f -name "*.sh" -exec chmod +x {} \; # Tornar executável
find . -type f -size 0 -delete                  # Deletar arquivos vazios
find . -type f -perm 777 -exec chmod 755 {} \;  # Corrigir permissões

# Find por data e tempo
find . -newermt "2023-01-01" -type f            # Arquivos mais novos que
data
find . -mtime +30 -type f                       # Modificados há mais de
30 dias
find . -atime -1 -type f                        # Acessados nas últimas
24h
find . -ctime -7 -type f                        # Criados nos últimos 7
dias

# Locate (mais rápido que find)
sudo updatedb                                   # Atualizar database
locate "*.conf" | grep nginx                    # Buscar arquivos
locate -i "blueprint"                           # Case insensitive
locate -c "*.log"                               # Contar resultados
locate -e "*.txt"                               # Apenas arquivos
existentes

# Which e whereis
which python3                                   # Localizar executável
whereis nginx                                   # Localizar binário,
source, manual
type -a python                                  # Todos os caminhos do
comando
command -v docker                               # Verificar se comando
existe
```

## 📝 Criação e Manipulação de Arquivos

```
# Criar arquivos com conteúdo
echo "Hello World" > file.txt                   # Sobrescrever
echo "New line" >> file.txt                     # Anexar
cat > file.txt << EOF                           # Heredoc
Line 1
Line 2
EOF

# Criar arquivos com tamanhos específicos
dd if=/dev/zero of=bigfile.txt bs=1M count=100  # 100MB de zeros
fallocate -l 1G largefile.txt                   # 1GB (mais rápido)
truncate -s 500M mediumfile.txt                 # 500MB vazio

# Criar múltiplos arquivos
touch file{1..10}.txt                           # file1.txt até file10.txt
touch {a,b,c}_{1,2,3}.txt                       # a_1.txt, a_2.txt, etc.
mkdir -p project/{src,docs,tests}/{js,css,html} # Estrutura complexa
```

```
# Templates de arquivos
cat > script_template.sh << 'EOF'
#!/bin/bash
set -euo pipefail

# Script: $1
# Author: $(whoami)
# Date: $(date)

main() {
    echo "Hello from $1"
}

main "$@"
EOF

# Criar arquivo com permissões específicas
install -m 755 /dev/null script.sh              # Criar executável
install -m 644 /dev/null config.conf            # Criar config
install -m 600 /dev/null secret.key             # Criar arquivo secreto
```

## 📂 Manipulação de Diretórios

```
# Criar estruturas complexas
mkdir -p
project/{frontend/{src/{components,pages,utils},public,tests},backend/{src/
{models,controllers,routes},tests},docs/{api,user}}

# Copiar estrutura de diretórios (sem arquivos)
find source_dir -type d -exec mkdir -p dest_dir/{} \;
rsync -av --include='*/' --exclude='*' source/ dest/

# Sincronizar diretórios
rsync -av --delete source/ dest/                # Sincronizar e deletar
extras
rsync -av --exclude='node_modules' src/ backup/ # Excluir diretórios
rsync -av --include='*.txt' --exclude='*' src/ dest/ # Apenas .txt

# Comparar diretórios
diff -r dir1 dir2                               # Diferenças recursivas
rsync -av --dry-run --delete src/ dest/         # Simular sincronização
find dir1 dir2 -name "*.txt" | sort | uniq -u   # Arquivos únicos

# Operações em lote
find . -type d -name "temp*" -exec rmdir {} \;  # Remover diretórios vazios
find . -type d -empty -delete                   # Deletar diretórios vazios
find . -name "*.bak" -exec mv {} backup/ \;     # Mover arquivos
```

## 🔧 Manipulação de Conteúdo

```
# Leitura avançada
head -n 20 file.txt                          # Primeiras 20 linhas
tail -n 50 file.txt                          # Últimas 50 linhas
tail -f file.log                             # Follow (tempo real)
tail -F file.log                             # Follow com retry
sed -n '10,20p' file.txt                     # Linhas 10 a 20
awk 'NR>=10 && NR<=20' file.txt              # Mesmo com awk

# Divisão de arquivos
split -l 1000 largefile.txt part_            # Dividir por linhas
split -b 10M largefile.txt part_             # Dividir por tamanho
csplit file.txt '/pattern/' '{*}'            # Dividir por padrão

# Junção de arquivos
cat part_* > arquivo_completo.txt            # Juntar partes
paste file1.txt file2.txt > combined.txt     # Colunas lado a lado
join file1.txt file2.txt > joined.txt        # Join por chave comum

# Ordenação e únicos
sort file.txt                                # Ordenar linhas
sort -n numbers.txt                          # Ordenação numérica
sort -k2 file.txt                            # Ordenar por coluna 2
sort -u file.txt                             # Ordenar e remover
duplicatas
uniq -c file.txt                             # Contar ocorrências
uniq -d file.txt                             # Apenas duplicatas
```

## 🔄 Transformação de Texto

```
# Substituições com sed
sed 's/old/new/g' file.txt                   # Substituir globalmente
sed 's/old/new/2' file.txt                   # Apenas 2ª ocorrência
sed '/pattern/d' file.txt                    # Deletar linhas com padrão
sed -i 's/old/new/g' file.txt                # Editar in-place
sed -i.bak 's/old/new/g' file.txt            # Editar com backup

# Transformações com tr
tr 'a-z' 'A-Z' < file.txt                    # Maiúsculas
tr -d '\r' < file.txt                        # Remover carriage return
tr -s ' ' < file.txt                         # Squeeze espaços
tr ' ' '\n' < file.txt                       # Espaços para newlines

# Processamento com awk
awk '{print $1}' file.txt                    # Primeira coluna
awk '{print NF}' file.txt                    # Número de campos
awk '{sum+=$1} END {print sum}' numbers.txt  # Somar primeira coluna
awk 'length($0) > 80' file.txt               # Linhas com mais de 80
chars
awk '!seen[$0]++' file.txt                   # Remover duplicatas
```

```
# Formatação de texto
column -t file.txt                          # Alinhar colunas
fmt -w 80 file.txt                          # Quebrar linhas em 80
chars
fold -w 60 file.txt                         # Quebrar palavras em 60
chars
expand file.txt                             # Tabs para espaços
unexpand file.txt                           # Espaços para tabs
```

## 🔐 Permissões e Propriedade

```
# Permissões numéricas
chmod 755 script.sh                         # rwxr-xr-x
chmod 644 config.txt                        # rw-r--r--
chmod 600 secret.key                        # rw-------
chmod 777 public_dir                        # rwxrwxrwx (cuidado!)

# Permissões simbólicas
chmod u+x script.sh                         # Adicionar execução para
owner
chmod g-w file.txt                          # Remover escrita para
grupo
chmod o-r secret.txt                        # Remover leitura para
outros
chmod a+r public.txt                        # Adicionar leitura para
todos

# Permissões recursivas
chmod -R 755 directory/                     # Recursivo
find . -type f -exec chmod 644 {} \;        # Apenas arquivos
find . -type d -exec chmod 755 {} \;        # Apenas diretórios

# Propriedade
chown user:group file.txt                   # Alterar owner e grupo
chown -R user:group directory/              # Recursivo
chgrp group file.txt                        # Apenas grupo
chown --reference=ref_file target_file      # Copiar propriedade

# Permissões especiais
chmod +t directory/                         # Sticky bit
chmod g+s directory/                        # SetGID
chmod u+s executable                        # SetUID
```

## 🔍 Análise de Arquivos

```
# Informações de arquivos
file arquivo.txt                            # Tipo do arquivo
stat arquivo.txt                            # Informações detalhadas
ls -la arquivo.txt                          # Permissões e tamanho
```

```
du -h arquivo.txt                          # Tamanho em disco
wc -l arquivo.txt                          # Contar linhas
wc -w arquivo.txt                          # Contar palavras
wc -c arquivo.txt                          # Contar caracteres

# Checksums e integridade
md5sum arquivo.txt                         # MD5 hash
sha256sum arquivo.txt                      # SHA256 hash
sha256sum arquivo.txt > arquivo.sha256     # Salvar hash
sha256sum -c arquivo.sha256                # Verificar integridade

# Comparação de arquivos
diff file1.txt file2.txt                   # Diferenças linha por
linha
diff -u file1.txt file2.txt                # Unified diff
diff -y file1.txt file2.txt                # Side by side
cmp file1.txt file2.txt                    # Comparação binária
comm file1.txt file2.txt                   # Linhas comuns/únicas

# Análise de conteúdo
grep -c "pattern" file.txt                 # Contar ocorrências
grep -n "pattern" file.txt                 # Com números de linha
grep -v "pattern" file.txt                 # Inverter match
grep -l "pattern" *.txt                    # Apenas nomes de arquivos
grep -r "pattern" directory/               # Busca recursiva
```

## 💾 Backup e Arquivamento

```
# Tar (arquivamento)
tar -czf backup.tar.gz directory/          # Criar arquivo comprimido
tar -xzf backup.tar.gz                     # Extrair arquivo
tar -tzf backup.tar.gz                     # Listar conteúdo
tar -czf backup.tar.gz --exclude='*.log' dir/   # Excluir arquivos

# Backup incremental
tar -czf full_backup.tar.gz directory/     # Backup completo
tar -czf incr_backup.tar.gz --newer-mtime="2023-01-01" directory/ #
Incremental

# Zip
zip -r backup.zip directory/               # Criar zip
unzip backup.zip                           # Extrair zip
unzip -l backup.zip                        # Listar conteúdo
zip -r backup.zip dir/ -x "*.log"          # Excluir arquivos

# Sincronização para backup
rsync -av --delete source/ backup/         # Backup espelho
rsync -av --backup --suffix=.bak src/ dest/     # Com backup de arquivos
alterados
rsync -av --link-dest=../previous current/ backup/ # Backup com hard links
```

## 🔧 Operações em Lote

```
# Renomeação em massa
rename 's/\.jpeg$/\.jpg/' *.jpeg                # Renomear extensões
rename 'y/A-Z/a-z/' *                           # Minúsculas
for file in *.txt; do mv "$file" "${file%.txt}.bak"; done # Bash loop

# Processamento em lote
find . -name "*.log" -exec gzip {} \;           # Comprimir logs
find . -name "*.txt" -exec sed -i 's/old/new/g' {} \; # Substituir em todos
parallel gzip ::: *.log                         # Comprimir em paralelo
(GNU parallel)

# Conversão de encoding
iconv -f ISO-8859-1 -t UTF-8 input.txt > output.txt # Converter encoding
dos2unix file.txt                               # Windows para Unix
unix2dos file.txt                               # Unix para Windows

# Operações condicionais
[ -f file.txt ] && echo "File exists"           # Se arquivo existe
[ -d directory ] || mkdir directory             # Criar se não existe
[ -r file.txt ] && cat file.txt                 # Ler se legível
[ -w file.txt ] && echo "data" >> file.txt      # Escrever se possível
```

## 📊 Monitoramento de Arquivos

```
# Monitorar mudanças
inotifywait -m -e modify,create,delete directory/ # Monitorar eventos
watch -n 1 'ls -la directory/'                  # Monitorar listagem
watch -n 5 'du -sh directory/'                  # Monitorar tamanho

# Encontrar arquivos grandes
find / -type f -size +100M 2>/dev/null          # Arquivos > 100MB
du -ah / | sort -rh | head -20                  # 20 maiores arquivos/dirs
ncdu /                                          # Interface interativa

# Arquivos duplicados
fdupes -r directory/                            # Encontrar duplicatas
rdfind directory/                               # Encontrar e gerenciar
duplicatas
find . -type f -exec md5sum {} \; | sort | uniq -d -w32 # Duplicatas por
hash

# Limpeza automática
find /tmp -type f -mtime +7 -delete             # Limpar arquivos antigos
find . -name "*.tmp" -delete                    # Limpar temporários
find . -type f -size 0 -delete                  # Limpar arquivos vazios
```

## 🚀 Scripts Úteis para Blueprint Blog

```
#!/bin/bash
# /opt/scripts/file-management.sh

# Backup de configurações
backup_configs() {
    local backup_dir="/backup/configs-$(date +%Y%m%d)"
    mkdir -p "$backup_dir"

    # Backup arquivos de configuração
    cp -r /etc/nginx "$backup_dir/"
    cp -r /opt/blueprint-blog/*.yml "$backup_dir/"
    cp /opt/blueprint-blog/.env "$backup_dir/"

    # Comprimir backup
    tar -czf "$backup_dir.tar.gz" "$backup_dir"
    rm -rf "$backup_dir"

    echo "Backup created: $backup_dir.tar.gz"
}

# Limpeza de logs antigos
cleanup_logs() {
    local days="${1:-30}"

    # Logs do sistema
    find /var/log -name "*.log" -mtime +$days -delete

    # Logs do Docker
    docker system prune -f

    # Logs do Blueprint Blog
    find /opt/blueprint-blog/logs -name "*.log" -mtime +$days -delete

    echo "Logs older than $days days cleaned"
}

# Organizar arquivos por data
organize_by_date() {
    local source_dir="$1"
    local dest_dir="$2"

    find "$source_dir" -type f -printf '%TY-%Tm-%Td %p\n' | while read date
file; do
        mkdir -p "$dest_dir/$date"
        mv "$file" "$dest_dir/$date/"
    done
}

# Verificar integridade de arquivos
check_integrity() {
    local dir="$1"
```

```
    # Gerar checksums
    find "$dir" -type f -exec sha256sum {} \; > "$dir/checksums.sha256"

    # Verificar checksums
    if sha256sum -c "$dir/checksums.sha256" --quiet; then
        echo "All files are intact"
    else
        echo "Some files are corrupted!"
        return 1
    fi
}

# Sincronizar com backup remoto
sync_remote_backup() {
    local local_dir="$1"
    local remote_host="$2"
    local remote_dir="$3"

    rsync -av --delete \
          --exclude='*.tmp' \
          --exclude='node_modules' \
          --exclude='target' \
          "$local_dir/" "$remote_host:$remote_dir/"
}

# Main function
case "${1:-}" in
    backup)
        backup_configs
        ;;
    cleanup)
        cleanup_logs "${2:-30}"
        ;;
    organize)
        organize_by_date "$2" "$3"
        ;;
    check)
        check_integrity "$2"
        ;;
    sync)
        sync_remote_backup "$2" "$3" "$4"
        ;;
    *)
        echo "Usage: $0 {backup|cleanup|organize|check|sync}"
        exit 1
        ;;
esac
```

alias ...='cd ../..' alias grep='grep --color=auto' alias fgrep='fgrep --color=auto' alias egrep='egrep --color=auto'

# Funções úteis

function mkcd() { mkdir -p "$1" && cd "$1"; } function backup() { cp "$1"{,.backup}; } function extract() { case $1 in _.tar.bz2) tar xjf $1 ;; _.tar.gz) tar xzf $1 ;; *.zip) unzip $1 ;; esac }

# Histórico inteligente

export HISTSIZE=10000 export HISTCONTROL=ignoredups:erasedups

```
### TypeScript Ninja

```bash
# Verificar tipos específicos
npx tsc --noEmit --listFiles | grep Component

# Gerar relatório de tipos
npx tsc --noEmit --pretty --listEmittedFiles

# Verificar compatibilidade de versões
npx tsc --showConfig | jq '.compilerOptions.target'

# Debug de resolução de módulos
npx tsc --traceResolution --noEmit src/main.tsx

# Verificar apenas arquivos modificados
git diff --name-only HEAD~1 | grep -E '\.(ts|tsx)$' | xargs npx tsc --
noEmit

# Análise de dependências circulares
npx madge --circular --extensions ts,tsx src/

# Verificar imports não utilizados
npx ts-unused-exports tsconfig.json
```

## Docker Ninja

```
# Limpeza completa
docker system prune -a --volumes
docker builder prune

# Inspecionar imagens
docker history blueprint-backend
docker inspect blueprint-backend

# Logs avançados
docker logs -f --tail 100 container_name
docker logs --since="2h" container_name

# Executar comandos em containers
docker exec -it container_name bash
```
```

```
docker exec container_name ls -la /app

# Multi-stage builds
docker build --target development .
docker build --target production .

# Docker Compose avançado
docker-compose up --scale backend=3
docker-compose logs -f backend
docker-compose exec backend bash

# Backup de volumes
docker run --rm -v myvolume:/data -v $(pwd):/backup alpine tar czf
/backup/backup.tar.gz /data
```

## Performance & Monitoring

```
# Monitorar uso de CPU/Memória
# Windows
Get-Process | Sort-Object CPU -Descending | Select-Object -First 10
Get-Counter "\Process(*)\% Processor Time"

# Linux
top -p $(pgrep node)
htop
iotop

# Monitorar rede
netstat -an | grep :3000
ss -tulpn | grep :3000

# Benchmark de build
time npm run build
Measure-Command { npm run build }

# Análise de bundle
npx webpack-bundle-analyzer dist/static/js/*.js
npx source-map-explorer dist/static/js/*.js

# Lighthouse CI
npm install -g @lhci/cli
lhci autorun
```

## Debugging Avançado

```
# Node.js debugging
node --inspect-brk=0.0.0.0:9229 node_modules/.bin/vite
node --inspect --inspect-port=9229 server.js
```

```
# Rust debugging
RUST_BACKTRACE=1 cargo run
RUST_BACKTRACE=full cargo run
RUST_LOG=debug,hyper=info cargo run

# Strace/ltrace (Linux)
strace -e trace=file cargo build
ltrace -e malloc cargo run

# Memory profiling
valgrind --tool=memcheck cargo run
```

## Automação & Scripts

```
# Git hooks
# .git/hooks/pre-commit
#!/bin/sh
npm run lint && npm run test

# Makefile para automação
.PHONY: dev build test clean
dev:
    npm run dev
build:
    npm run build
test:
    npm test && cargo test
clean:
    rm -rf node_modules target dist

# Scripts de deploy
#!/bin/bash
set -e
npm run build
docker build -t app:latest .
docker push registry.com/app:latest
kubectl apply -f k8s/
```

## Truques de Produtividade

```
# Aliases para projetos
alias blog='cd ~/projects/blueprint-blog'
alias be='cd ~/projects/blueprint-blog/backend'
alias fe='cd ~/projects/blueprint-blog/frontend'

# Funções para desenvolvimento
function dev() {
    cd ~/projects/blueprint-blog
    code .
```

```
    npm run dev
}

function deploy() {
    git add .
    git commit -m "${1:-Update}"
    git push
    npm run build
}

# Tmux/Screen para sessões persistentes
tmux new-session -d -s blog
tmux send-keys -t blog 'cd ~/projects/blueprint-blog && npm run dev' Enter

# Watchdog para restart automático
nodemon --watch src --ext ts,tsx --exec "npm run build"
cargo watch -x run
```

## Troubleshooting Ninja

```
# Verificar locks de arquivos
lsof +D /path/to/directory
fuser -v /path/to/file

# Verificar processos zumbis
ps aux | grep -E "(Z|<defunct>)"

# Limpar DNS cache
# Windows
ipconfig /flushdns
# Linux
sudo systemctl flush-dns

# Verificar conectividade
curl -I http://localhost:3000
telnet localhost 3000
nc -zv localhost 3000

# Verificar certificados SSL
openssl s_client -connect domain.com:443
curl -vI https://domain.com

# Análise de logs
tail -f /var/log/nginx/error.log | grep -E "(error|warning)"
journalctl -u nginx -f --since "1 hour ago"
```

---

*📝 Documento criado durante a migração TypeScript 100% do Blueprint Blog 🗓️ Data: Janeiro 2025 🥷 Comandos ninja adicionados para máxima produtividade!*