FreeCORE

THE COMPLETE GUIDE

DEPLOYING, ADMINISTERING, AND MASTERING THE OPEN-SOURCE NAS PLATFORM BORN FROM TRUENAS CORE

POWERED BY
BUILT ON FREEBSD & OPENZFS
FREEBSD & OPENZFS

EVERYTHING YOU NEED TO BUILD, MANAGE, AND PROTECT YOUR DATA

- ZFS STORAGE FUNDAMENTALS
- NETWORK SHARING PROTOCOLS
- SECURITY HARDENING
- BACKUP & DISASTER RECOVERY
- PERFORMANCE TUNING
- TROUBLESHOOTING & RECOVERY

- Step-by-step instructions with real-world examples
- Verification steps to ensure success
- Recovery guidance when things go wrong

PRACTICAL.
RELIABLE.
PRODUCTION READY.

THOMAS COLEMAN

FreeCORE - The Complete Guide

Deploying, Administering, and Mastering the Open-Source NAS Platform Born from TrueNAS CORE

Steve Publications

This book is available at https://leanpub.com/freecore-thecompleteguide

This version was published on 2026-08-30

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Deploying, Administering, and Mastering the Open-Source NAS Platform Born from TrueNAS CORE

FreeCORE carries forward the proven architecture of TrueNAS CORE as an independently maintained storage operating system built on FreeBSD and OpenZFS. This book is your comprehensive reference for deploying, configuring, and administering FreeCORE in environments ranging from home labs to production data centers. You will learn the fundamentals of ZFS storage pools, network sharing protocols, security hardening, backup strategies, performance tuning, and systematic troubleshooting. Every procedure includes realistic examples, verification steps, and recovery guidance so you can operate FreeCORE with confidence and keep your data safe.

Introduction

You have storage to protect. Perhaps it is family photos that cannot be replaced, or business records that must survive hardware failures, or virtual machine disks that need reliable block storage. Whatever your use case, the core requirement is the same: you need a system that keeps data intact, available, and recoverable without demanding constant babysitting.

FreeCORE exists for exactly this purpose. It is a continuation of TrueNAS CORE, the FreeBSD-based network-attached storage platform that grew from FreeNAS into one of the most widely deployed open-source storage solutions in the world. When iXsystems ended active development of TrueNAS CORE and shifted focus to its Linux-based SCALE variant, the community responded with FreeCORE: an independent fork that takes TrueNAS CORE 13.3 forward on a modern FreeBSD 15 base while preserving the architecture, workflow, and reliability that administrators came to trust.[1][2]

This book assumes you are serious about your data. It is written for system administrators who need production-grade storage, IT professionals evaluating NAS platforms, homelab enthusiasts building reliable personal infrastructure, and anyone who wants to understand not just how to configure FreeCORE but why each decision matters. You will find step-by-step procedures for installation and configuration, deep explanations of ZFS concepts that underlie every storage operation, practical guidance on sharing data over SMB, NFS, iSCSI, and other protocols, and systematic approaches to security hardening, backup, performance tuning, and troubleshooting.

What This Book Covers

The chapters progress from foundation to mastery:

Chapter 1 tells the story of FreeCORE: how TrueNAS evolved from FreeNAS, why iXsystems ended free support for its FreeBSD variant, and what FreeCORE preserves versus what it changes. Understanding this history helps you evaluate whether FreeCORE fits your environment.

Chapters 2 and 3 explain architecture and planning. You will learn how FreeBSD and OpenZFS integrate to create a storage stack that prioritizes data integrity above all else, and how to plan hardware selections, pool topologies, and network designs before installing anything.

Chapters 4 and 5 walk through installation and initial configuration with complete, step-by-step procedures. You will install FreeCORE on real hardware, configure networking, create your first storage pool, set up users and services, and verify that everything works.

Chapter 6 is a deep dive into ZFS concepts: copy-on-write semantics, checksumming, vdevs, RAID-Z layouts, snapshots, replication, and the caching mechanisms that make spinning disks feel fast. This chapter does not just describe features; it explains the mechanisms so you can reason about trade-offs.

Chapters 7 through 12 cover day-to-day administration: managing pools and disks, implementing snapshot and replication strategies, configuring networking including LAGG and VLANs, deploying file sharing services, integrating with Active Directory and LDAP, hardening security, setting up monitoring, and performing routine maintenance.

Chapter 13 brings it all together with advanced topics: migrating from TrueNAS CORE to FreeCORE, automating administration via CLI and API, tuning performance for specific workloads, and systematic troubleshooting when things go wrong.

How to Use This Book

Read sequentially if you are new to FreeCORE or ZFS. The early chapters build foundational knowledge that later chapters assume. If you already know TrueNAS CORE or FreeBSD/ZFS, skim the history and architecture chapters and jump directly to the procedures relevant to your task.

Throughout the book you will encounter:

- **Procedures**: Step-by-step instructions for specific tasks, each with verification steps showing how to confirm success.
- **Examples**: Realistic command sequences, configuration values, and deployment scenarios that you can adapt to your environment.

- **Warnings**: Clearly marked alerts about operations that destroy data, cause downtime, or create security risks.
- **FreeCORE Notes**: Callouts distinguishing FreeCORE-specific behavior from legacy TrueNAS CORE functionality where differences matter.

All procedures are written for FreeCORE 15.0, the first stable release based on FreeBSD 15 and OpenZFS.[3] Where functionality is inherited directly from TrueNAS CORE without modification, this book notes that inheritance explicitly rather than repeating identical content. Where information about FreeCORE is incomplete or evolving, I state that uncertainty directly instead of filling gaps with assumptions.

Prerequisites

This book assumes:

- Basic familiarity with Unix-style command-line interfaces
- Understanding of fundamental networking concepts (IP addressing, subnets, DNS)
- Comfort reading technical documentation and following multi-step procedures
- Access to hardware suitable for NAS deployment (covered in Chapter 3)

No prior ZFS or TrueNAS experience is required. Every major concept is explained from first principles before being applied in practice.

Version Notes

FreeCORE 15.0 was released on August 21, 2026 as the project's first stable version.[4] It runs on FreeBSD 15.1-RELEASE and supports in-place upgrades from TrueNAS CORE 13.3 systems.[5] This book documents FreeCORE 15.0 behavior wherever possible, drawing on official FreeCORE documentation and the underlying FreeBSD and OpenZFS references when FreeCORE-specific details are not yet published separately.

If you are reading this book after a newer FreeCORE version has been released, compare your version's release notes against the procedures here.

Core ZFS operations and FreeBSD fundamentals rarely change between releases, but interface locations, default settings, and supported features may shift. When in doubt, consult the official FreeCORE documentation at https://docs.freecore.org/ for the most current information.

Let us begin with the story of how FreeCORE came to exist and why it matters.

Chapter 1: The Story of FreeCORE, From TrueNAS CORE to Open-Source Continuation

The TrueNAS Legacy: How an Open-Source NAS Became Industry Standard

The lineage of FreeCORE stretches back more than two decades, through several names and strategic pivots that shaped what has become one of the most widely deployed open-source storage platforms in existence.

FreeNAS began in October 2005 as a project by Olivier Cochard-Labbé, based on FreeBSD and designed to turn commodity hardware into network-attached storage.[6] Early versions were modest, offering basic file sharing over SMB and NFS with a web-based management interface. What set FreeNAS apart from competing solutions was its focus on simplicity: an administrator could install the software on a USB stick, boot it into RAM, add some hard drives, and have a working NAS within minutes.

The defining moment for FreeNAS arrived in 2010 when the project rebased from Linux to FreeBSD, bringing with it access to OpenZFS.[7] This decision was transformative. ZFS, originally developed at Sun Microsystems by Jeff Bonwick, Matthew Ahrens, and Bill Moore starting in 2001, represented a fundamental rethink of how storage systems should work.[8][9] Unlike traditional filesystems layered on top of separate RAID controllers, ZFS combined volume management and filesystem roles into a single coherent stack that treated physical disks as a pool of capacity while protecting data integrity through end-to-end checksumming.

FreeNAS 8, released in 2013 with ZFS at its core, became the version that many administrators still remember fondly: stable, straightforward, and genuinely open-source under the BSD license. It ran on FreeBSD, supported jails for running additional services, offered bhyve virtualization, and gave

users complete control over their storage configuration through a clean web interface.

iXsystems, founded in 1996 as one of the earliest commercial FreeBSD companies, became closely associated with FreeNAS over time. While FreeNAS remained community-developed and open-source, iXsystems sold hardware appliances preloaded with the software and offered paid support contracts. In 2015, Kris Moore, founder of PC-BSD (later TrueOS), joined iXsystems and led a major redesign that became FreeNAS 9.x, introducing a modernized interface based on Django and AngularJS.

The rebranding to TrueNAS occurred in March 2020. FreeNAS was renamed TrueNAS CORE, while iXsystems' enterprise appliance line became TrueNAS Enterprise.[10] The transition was largely cosmetic for community users: TrueNAS CORE retained the same FreeBSD base, the same ZFS storage engine, and the same open-source licensing. Under the hood, however, the project had shifted toward an "open core" model where advanced features such as high availability clustering and Fibre Channel support were available only with paid enterprise licenses.

By 2023, TrueNAS CORE had reached version 13.x running on FreeBSD 13, supporting OpenZFS 2.x with modern features including native encryption at the dataset level, improved compression algorithms, and enhanced snapshot management. It remained one of the most popular choices for homelab storage and small-to-medium business NAS deployments worldwide.

The Fork That Changed Everything: iXsystems Ends Free TrueNAS CORE Support

The trajectory changed in early 2024 when iXsystems announced that TrueNAS CORE 13.3 would be the last FreeBSD-based release.[11][12] The company had been developing TrueNAS SCALE, a Linux-based variant built on Debian with Docker and Kubernetes integration, since 2020.[13] SCALE represented iXsystems' strategic future: container-native, horizontally scalable, and aligned with the broader industry shift toward Linux-based infrastructure.

The announcement was clear but understated. There would be no TrueNAS CORE 14. The FreeBSD line would stop at 13.3, released in June 2024.[14] iXsystems described this as an "extended maintenance mode" rather than immediate abandonment, but the message to community users was unmistakable:

FreeBSD-based TrueNAS CORE had reached its end of life from a development perspective.

The implications were serious for existing deployments. FreeBSD 13.x itself would reach end-of-life in July 2024, meaning security patches and bug fixes would cease flowing upstream.[15] Without iXsystems maintaining the TrueNAS-specific code on top of FreeBSD, community users faced a choice: migrate to TrueNAS SCALE with its different OS base and feature set, move to an alternative NAS platform entirely, or attempt to maintain their own fork.

The reaction in the community was swift and divided. Some administrators embraced the migration to SCALE, attracted by native Docker support and modern container orchestration. Others were reluctant to leave FreeBSD: they valued its stability, its mature ZFS integration, and the fact that their existing deployments had been running reliably for years without reason to change. A vocal segment of the community expressed frustration with what they perceived as a "rug pull" on an open-source project that had built its reputation on community trust.[16]

The licensing situation added complexity. TrueNAS CORE was released under the BSD license, which permits forking and independent redistribution. However, iXsystems controlled the build infrastructure, update servers, and much of the middleware code that made TrueNAS usable as a complete product rather than raw FreeBSD with ZFS installed. Community members who wanted to continue developing on FreeBSD would need to recreate or fork not just the operating system base but the entire management layer.

Enter FreeCORE: Community, Philosophy, and Project Goals

FreeCORE emerged as a direct response to this situation. Launched in August 2026, it is an independent fork of TrueNAS CORE 13.3 that rebases the project onto FreeBSD 15 while maintaining compatibility with existing deployments.[1][2] The FreeCORE Project Team describes their mission straightforwardly: carrying forward TrueNAS CORE as a truly open-source operating system, unencumbered by commercial licensing restrictions and independent of any single corporate sponsor.[17]

The project's technical approach is pragmatic. Rather than rebuilding the entire TrueNAS stack from scratch, FreeCORE takes the proven TrueNAS CORE

13.3 codebase and applies it to a modern FreeBSD foundation. This strategy preserves three critical things:

First, existing storage pools remain compatible. A TrueNAS CORE 13.3 pool can be imported directly into FreeCORE without migration or data movement. The ZFS filesystem layer is unchanged; only the operating system base and management middleware are updated.

Second, administrators retain their workflows. The web interface, command-line tools, configuration procedures, and API endpoints are familiar to anyone who has used TrueNAS CORE. Learning curves and operational disruptions are minimized.

Third, jails and bhyve virtualization continue to work. Unlike the migration path from TrueNAS CORE to TrueNAS SCALE, which requires recreating FreeBSD jails as Linux containers or migrating VMs entirely, FreeCORE maintains native FreeBSD virtualization capabilities.[18]

FreeCORE 15.0, released on August 21, 2026, runs on FreeBSD 15.1-RELEASE and includes several notable changes from TrueNAS CORE 13.3.[4][19] OpenVPN has been replaced with WireGuard for VPN functionality. Enterprise-only features that required iXsystems license keys (high availability failover, KMIP key management, Fibre Channel support, enclosure management, and TrueCommand integration) have been removed entirely rather than left as non-functional placeholders. Vendor telemetry and the host MinIO server package were also stripped out. Conversely, FreeCORE added new capabilities: WebAuthn security key support for two-factor authentication, console OTP authentication, the rar2fs service for mounting RAR archives as filesystems, and updated graphics drivers.

The project operates openly on Codeberg, a decentralized Git hosting platform chosen to avoid dependency on any single corporate provider.[20] Source code is available under permissive open-source licenses consistent with the FreeBSD and TrueNAS CORE tradition. The team communicates via email (hello@freecore.org for general inquiries, security@freecore.org for vulnerability reports), IRC (#freecore on Libera.Chat), and community forums.

FreeCORE explicitly states it is not affiliated with iXsystems or The FreeBSD Foundation.[1] This independence is both a strength and a consideration: the project answers to its users rather than a corporate strategy, but it also lacks the financial resources and full-time engineering staff that iXsystems brings to TrueNAS development. For now, FreeCORE is maintained by volunteers

and community contributors who believe in preserving FreeBSD-based open-source NAS as a viable option.

What's the Same, What's Different: Comparing FreeCORE to TrueNAS CORE

Understanding where FreeCORE diverges from TrueNAS CORE matters for operational planning. The following comparison covers key areas where behavior is identical versus changed.

Operating system base: TrueNAS CORE 13.3 runs on FreeBSD 13.x; FreeCORE 15.0 runs on FreeBSD 15.1-RELEASE.[2][4] This is the most significant change and brings kernel improvements, updated drivers, newer ports, and security patches that FreeBSD 13.x users no longer receive.

ZFS version: Both use OpenZFS from the FreeBSD base. FreeCORE benefits from whatever ZFS features and fixes are included in FreeBSD 15.1. Core ZFS behavior (pool management, datasets, snapshots, replication, checksumming) is functionally identical between the two platforms when running comparable OpenZFS versions.

Web interface: The FreeCORE web UI is derived from TrueNAS CORE's interface with modifications for FreeBSD 15 compatibility and removal of enterprise-only features. Navigation patterns, configuration screens, and terminology are largely preserved. Some visual elements and default settings may differ due to the redesigned web Shell and updated middleware.[4]

Storage pools: Existing TrueNAS CORE 13.3 ZFS pools are directly importable into FreeCORE without modification. Pool topology, dataset structure, snapshots, replication tasks, and encryption keys transfer intact during an in-place upgrade.[5]

File sharing services: SMB (via Samba), NFS, iSCSI, FTP/SFTP, AFP, WebDAV, and Rsync are all supported in FreeCORE using the same configuration mechanisms as TrueNAS CORE.[21][22] Protocol versions and capability sets depend on the underlying FreeBSD ports rather than custom TrueNAS code.

Virtualization: FreeBSD jails (managed via iocage) and bhyve virtual machines are maintained in FreeCORE.[4] This is a critical difference from migrating to TrueNAS SCALE, where FreeBSD jails cannot run natively and must be replaced with Linux containers or VMs.

Directory services: Active Directory integration, LDAP authentication, and Kerberos configuration follow the same patterns as TrueNAS CORE.[23] The underlying PAM and NSS mechanisms are FreeBSD-native rather than TrueNAS-customized.

Networking: LAGG link aggregation, VLAN tagging, bridging, DHCP client/server, DNS resolution, and firewalling via PF are all supported with familiar configuration interfaces.[24][25] FreeBSD 15 networking stack improvements may affect driver support for specific NICs.

Removed features: Enterprise-only capabilities absent from TrueNAS CORE community edition remain absent in FreeCORE. Additionally, OpenVPN is replaced by WireGuard; the host MinIO server package is removed; and vendor telemetry collection has been eliminated.[4]

Update mechanism: FreeCORE maintains its own update infrastructure at updates.freecore.org, independent of iXsystems' TrueNAS update servers.[5] The boot environment upgrade model inherited from TrueNAS CORE (creating a new bootable clone before switching) remains unchanged.

Choosing Your Path: FreeCORE vs. TrueNAS SCALE vs. Alternatives

If you are running TrueNAS CORE today and evaluating options, the decision tree depends on your priorities. FreeCORE is not universally the right choice; it solves specific problems for specific users.

Choose FreeCORE if: You are running TrueNAS CORE 13.3 and want to continue using FreeBSD without migrating to a different operating system. Your existing jails, VMs, and workflows depend on FreeBSD compatibility. You value open-source independence and do not require enterprise features like HA clustering or Fibre Channel. You are comfortable with a community-maintained project that may move slower than a corporate-backed alternative.

Choose TrueNAS SCALE if: You want native Docker and Kubernetes integration for running modern containerized applications. Your organization already standardizes on Linux infrastructure. You need iXsystems commercial support contracts. You are starting a new deployment rather than migrating an existing FreeBSD-based system. Be aware that migrating from CORE to SCALE is one-way and requires recreating jails as containers or VMs.[26]

Choose vanilla FreeBSD with ZFS if: You want maximum control over your storage stack without the TrueNAS middleware layer. You are comfortable configuring Samba, NFS, networking, and services manually via configuration files rather than a web interface. You need to customize every aspect of the operating system for specialized workloads. This option requires significantly more administrative overhead but offers complete flexibility.

Consider alternatives if: Your needs do not align with ZFS-based storage (for example, you prefer btrfs on Linux or are constrained by specific hardware compatibility requirements). Projects like OpenMediaVault (Linux/Debian based), XigmaNAS (FreeBSD fork with different feature priorities), and commercial appliances from Synology or QNAP may better fit certain use cases.

For the purposes of this book, we assume you have decided that FreeCORE is the right platform for your environment. The remaining chapters provide everything you need to deploy it correctly, administer it effectively, and keep your data protected over the long term.

Chapter 2: Architecture Deep Dive, Understanding How FreeCORE Works

FreeBSD at the Core: Why This Matters for Stability and Performance

FreeCORE is FreeBSD first and NAS platform second. Every storage operation, network connection, authentication decision, and service daemon runs on top of the FreeBSD operating system kernel and userspace. Understanding this foundation explains why FreeCORE behaves the way it does and why certain design choices were made.

FreeBSD is a Unix-derived operating system descended from BSD (Berkeley Software Distribution) at UC Berkeley in the 1970s. It has been developed continuously since 1979 and is known for several characteristics that make it particularly well-suited to storage workloads:

Kernel stability: FreeBSD follows a conservative release model. Each major version undergoes extended testing before release, and maintenance branches receive security patches and critical bug fixes for years. FreeBSD 15.1-RELEASE, the base for FreeCORE 15.0, represents the culmination of this approach.[4] For storage systems where unexpected kernel panics can mean data unavailability or corruption, this stability matters.

Memory management: FreeBSD's virtual memory system is mature and efficient. It handles large memory configurations gracefully and provides predictable behavior under sustained workloads , essential for ZFS, which uses available RAM aggressively for caching (the ARC, discussed later in this chapter).

Networking stack: FreeBSD's TCP/IP implementation is widely regarded as one of the best in the industry. It includes advanced features like TCP window scaling, selective acknowledgments, SYN cookies for denial-of-service protection, and efficient handling of high packet rates. These capabilities matter when your NAS serves files to dozens or hundreds of clients simultaneously over gigabit or 10-gigabit networks.

Device drivers: FreeBSD supports a broad range of hardware through its in-tree driver collection. For storage systems, this includes SCSI/SAS controllers (LSI/Broadcom MegaRAID in IT mode, Adaptec, Areca), SATA controllers, NVMe devices, and Ethernet NICs from Intel, Broadcom, Realtek, and Mellanox. Driver quality varies by vendor and device generation, so hardware compatibility should always be verified before purchase.

Jails: FreeBSD jails provide operating-system-level virtualization that predates Linux containers by decades.[27] A jail is an isolated execution environment with its own filesystem tree, network stack (via VNET), process namespace, and user accounts, all sharing the host kernel. FreeCORE uses jails to run additional services like Samba, NFS server daemons, or custom applications in isolation from the core operating system.

Package management: FreeBSD's pkg system provides binary packages built from the ports tree, offering a curated collection of software that can be installed and updated reliably. FreeCORE leverages this infrastructure to include services like Samba, OpenSSH, SNMP, and other NAS-relevant daemons without compiling everything from source.

The choice of FreeBSD over Linux for FreeCORE is deliberate. While Linux has broader hardware support and larger community ecosystems, FreeBSD offers several advantages specifically relevant to storage: tighter ZFS integration (OpenZFS on FreeBSD is maintained as part of the base system rather than as an out-of-tree module), a more conservative approach to changing established APIs, and a development culture that prioritizes correctness and stability over feature velocity.

OpenZFS Integration: The Storage Engine That Makes It All Work

If FreeBSD provides the foundation, OpenZFS is the engine. ZFS is not merely a filesystem layered on top of block devices; it is a unified storage stack that combines volume management, filesystem operations, RAID functionality, and data integrity protection into a single coherent system. This architectural decision eliminates many failure modes present in traditional storage stacks while introducing new concepts that administrators must understand.

In a conventional Linux or Windows storage setup, the stack looks like this: physical disks are managed by a hardware RAID controller or software RAID

layer (mdadm, LVM), which presents logical volumes to the operating system, which then formats those volumes with a filesystem (ext4, xfs, NTFS). Each layer operates independently, and data integrity protection is typically limited to what the filesystem provides , usually simple checksums on metadata but not on user data.

ZFS collapses this stack. Physical disks are added directly to a ZFS pool without any intermediate RAID or volume manager. The pool presents capacity as datasets (filesystem-like containers) or zvols (block devices). Every block of data written to the pool is checksummed, and every read verifies that checksum against the stored value. If corruption is detected and redundant copies exist (via mirrors or RAID-Z parity), ZFS automatically repairs the damaged block using the good copy.

The OpenZFS implementation in FreeBSD 15 is integrated directly into the kernel rather than loaded as an external module.[28] This integration provides several benefits:

Boot-from-ZFS support: FreeBSD can boot from a ZFS pool, which FreeCORE uses for its system dataset and boot environments. The bootloader (either UEFI or legacy BIOS) loads the FreeBSD kernel, which then imports the root ZFS pool and continues booting , all without requiring separate partition management.

Consistent updates: When FreeBSD releases security patches or feature updates that affect ZFS behavior, they are delivered together as part of the operating system release rather than requiring coordination between separate package maintainers.

Native tooling: ZFS administration commands (zpool, zfs) and related utilities are included in the base system. Monitoring tools like arcstat and zdb are available without additional installation.

FreeCORE does not reimplement ZFS; it relies entirely on FreeBSD's OpenZFS implementation. This means that ZFS behavior in FreeCORE matches FreeBSD documentation for the corresponding version. Any ZFS feature, property, or limitation documented for FreeBSD 15.1 with its included OpenZFS version applies directly to FreeCORE.

The practical implication is that learning ZFS administration , pool creation, dataset management, snapshot operations, replication procedures, scrub scheduling, and troubleshooting , is not platform-specific knowledge. Skills you develop administering ZFS on FreeBSD apply equally to FreeCORE,

TrueNAS CORE, Proxmox with ZFS root, or any other system using OpenZFS on FreeBSD.

The Middleware Layer: How Services Talk to the Operating System

Between the FreeBSD kernel (with ZFS) and the user-facing interfaces sits a middleware layer that translates administrative actions into operating system operations. This layer is inherited from TrueNAS CORE with modifications for FreeBSD 15 compatibility and FreeCORE-specific changes.

The middleware is implemented primarily in Python and provides several critical functions:

Configuration management: Every setting changed through the web interface , network configuration, share definitions, user accounts, service parameters , is stored in a SQLite database managed by the middleware. When you create an SMB share through the UI, the middleware writes the share definition to its database, generates the appropriate Samba configuration files, and restarts the Samba service with the new settings.

Service orchestration: FreeCORE services (SMB, NFS, iSCSI target, FTP, SSH, SNMP, etc.) are managed by the middleware rather than directly through FreeBSD's rc system. Each service has a corresponding middleware module that handles startup, shutdown, status reporting, and configuration generation. This abstraction allows the web interface to control services uniformly regardless of underlying implementation details.

REST API: The middleware exposes a RESTful API that provides programmatic access to all configuration operations available through the web UI. Every action you can perform in the browser , creating pools, adding users, configuring shares, scheduling tasks , has a corresponding API endpoint. This API enables automation via scripts, integration with external management tools, and potential third-party client applications. The API uses HTTP/HTTPS with token-based authentication and returns JSON responses.[29]

Task scheduler: Scheduled operations like periodic snapshots, replication tasks, scrubs, and SMART tests are managed by middleware schedulers that create FreeBSD cron entries or use the system's task queue. The middleware tracks task execution history, success/failure status, and output logs.

Web interface backend: The Django-based web application communicates with the middleware through internal APIs to display system state and accept configuration changes. The frontend (JavaScript/HTML/CSS) renders the user interface, while the backend validates inputs, enforces constraints, and delegates operations to appropriate middleware modules.

Understanding this architecture helps explain certain behaviors:

- Configuration changes may take a moment to apply because the middleware must update its database, regenerate configuration files, and restart affected services.
- The web UI and API are the canonical interfaces for configuration. While you can modify FreeBSD configuration files directly (sshd_config, pf.conf, etc.), those changes may be overwritten by the middleware on service restart or system update unless made through the proper channels.
- Service status shown in the web interface reflects what the middleware knows about each daemon, not necessarily direct process inspection. If a service crashes unexpectedly, the middleware may still report it as running until its next status check cycle.

The Web UI and CLI: Two Interfaces, One System

FreeCORE provides two primary interfaces for administration: the web-based graphical user interface and the command-line shell. Both access the same underlying system and configuration database; they are different windows into the same functionality.

Web interface: Accessed via HTTPS on port 8000 by default (the web UI binds to all network interfaces), the FreeCORE web interface provides point-and-click administration for virtually every aspect of the system. It is organized into logical sections: Dashboard (system overview and alerts), Storage (pools, datasets, snapshots), Sharing (SMB, NFS, iSCSI, WebDAV shares), Tasks (scheduled operations), Credentials (users, groups, directory services), Network (interfaces, VLANs, routing), Services (enabled daemons), System (settings, updates, boot environments), and Shell (web-based terminal).

The web interface is designed for most day-to-day administration. It provides validation to prevent invalid configurations, wizards that guide complex setup procedures step-by-step, visual representations of pool topology and

disk status, and integrated documentation links. For administrators managing multiple systems or performing repetitive tasks, the web UI's consistency reduces cognitive load compared to remembering command syntax across different tools.

Command-line interface: FreeCORE provides SSH access (port 22 by default) with a shell prompt running on FreeBSD. From the shell, you have access to:

- Standard FreeBSD utilities: ps, top, vmstat, netstat, ifconfig, route, pkg, etc.
- ZFS administration commands: zpool, zfs, zdb, arc_summary
- TrueNAS/FreeCORE-specific tools: midclt (middleware client for API calls from the command line), bectl (boot environment management)
- Service control via the middleware: service commands that interact with FreeCORE's service management layer rather than raw FreeBSD rc scripts

The CLI is essential for several scenarios: troubleshooting when the web interface is inaccessible, performing operations not yet exposed through the UI, scripting automation tasks, examining low-level system state (kernel logs, ZFS internals), and connecting to systems from remote locations where a browser is impractical.

A key principle: both interfaces modify the same configuration database. If you create a user through the web UI, that user appears in FreeBSD's password file and is accessible via SSH. If you run zfs snapshot from the CLI, that snapshot appears in the Storage > Snapshots screen. The two interfaces are not competing systems; they are complementary access methods to one unified platform.

Boot Environments and Updates: Atomic Upgrades Explained

One of FreeBSD's most valuable features for production systems is boot environments (BEs), and FreeCORE inherits this capability directly. A boot environment is a complete, bootable copy of the operating system that can be activated or deactivated without affecting other environments. This mechanism enables safe upgrades with guaranteed rollback if something goes wrong.

Here is how it works in practice:

When you install FreeCORE initially, the installer creates a boot pool (typically on the installation USB drive or a small SSD) containing the base FreeBSD system, TrueNAS middleware, and configuration database. This initial boot environment is named something like "FreeCORE-15.0-U1" and is marked as active.

When an update becomes available (say, FreeCORE 15.0-U2), the upgrade process does not modify the existing boot environment in place. Instead, it creates a new boot environment by cloning the current one at the ZFS dataset level (a fast, space-efficient operation that shares unchanged blocks with the original). The new environment is then updated with the new system files, middleware changes, and any database schema migrations required. Once the new environment is fully prepared and validated, the bootloader configuration is updated to make it the default on next reboot.

When you reboot after an update, the system boots from the new environment. If everything works correctly, you continue operating on the updated version. If something is broken (a kernel panic, a middleware crash, an incompatible change that breaks your services), you can boot into the previous environment by selecting it from the bootloader menu or using bectl to activate it remotely.

This process provides several critical safety properties:

- **Atomicity:** An upgrade either completes fully (new BE is created and marked active) or fails without partially modifying the running system. There is no state where half the system is old and half is new.
- **Rollback capability:** Previous boot environments remain available until explicitly deleted. You can return to a known-good version within minutes rather than reinstalling from media.
- **Space efficiency:** ZFS clones share blocks with their parent dataset. A new boot environment initially consumes almost no additional space beyond metadata; it grows only as files are modified in the cloned copy.

FreeCORE manages boot environments through both the web interface (System > Boot Environments) and the bectl command-line tool. Best practices include:

- Keeping at least one previous boot environment after any major update until you have verified system stability for several days

- Deleting old environments periodically to prevent the boot pool from filling up (the installer creates a relatively small boot pool, typically 8-16 GB)
- Never modifying files in a boot environment directly via CLI while it is not active; changes may be lost when that environment is deleted

The update mechanism applies to both major version upgrades (when FreeCORE releases new versions like 15.0 to a hypothetical future 16.0) and maintenance updates (U1, U2, etc., within the same major version). The boot environment model ensures that all upgrade paths maintain rollback capability regardless of scope.

Understanding boot environments is not just theoretical knowledge; it is practical insurance for your production system. When you perform any update on FreeCORE, you are relying on this mechanism to protect you from upgrade failures. Knowing how it works helps you use it correctly and recover confidently when problems occur.

Chapter 3: Planning Your Deployment, Requirements, Topologies, and Design Decisions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Hardware Requirements: Minimum, Recommended, and Production-Class

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

CPU, Memory, and Motherboard: What Matters for ZFS Performance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Storage Planning: Disk Selection, Controllers, and Topology Choices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Network Design: Speed, Redundancy, and Protocol Considerations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Deployment Scenarios: Home Lab, Small Business, Enterprise Edge

Chapter 4: Installation, Getting FreeCORE Running on Your Hardware

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Preparing Your Installation Media: USB Creation and Verification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

The Installer Walkthrough: Every Screen Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Advanced Installation Options: Disk Selection and Boot Configuration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Verifying the Installation: Health Checks and First Boot Procedures

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Troubleshooting Installation Problems: Common Failure Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Chapter 5: Initial Configuration, From Fresh Install to Functional NAS

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Network Configuration: IP Addressing, DNS, and Gateway Setup

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

System Settings: Hostname, Time Synchronization, and Console Access

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Creating Your First Storage Pool: Step-by-Step with Verification

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Initial User and Group Setup: Local Accounts and Privileges

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Enabling Essential Services: SSH, NTP, and Logging

Chapter 6: ZFS Fundamentals, Understanding the Filesystem That Powers Everything

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Why ZFS: Copy-on-Write, Checksumming, and Self-Healing Explained

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Pool Architecture: How vdevs, Mirrors, and RAID-Z Work Together

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Datasets and Volume Types: Organizing Your Storage Logically

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

RAID Configurations Compared: Mirror vs. RAIDZ1 vs. RAIDZ2 vs. RAIDZ3

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

The ZFS Intent Log and Write Cache: Performance and Safety Trade-offs

Chapter 7: Storage Administration, Managing Pools, Datasets, and Disks

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Creating and Expanding Pools: Procedures and Best Practices

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Dataset Management: Properties, Quotas, and Reservations

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Disk Replacement Procedures: From Failure Detection to Rebuild Completion

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Scrubs: Scheduling, Running, and Interpreting Results

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

SMART Monitoring and Predictive Failure Analysis

Chapter 8: Snapshots, Replication, and Data Protection

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Snapshot Fundamentals: Creating, Managing, and Restoring Point-in-Time Copies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Automated Snapshot Policies: Schedules, Retention, and Naming Conventions

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Replication Basics: Push vs. Pull, Deduplication, and Bandwidth Limits

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Building a Backup Strategy with Snapshots and Replication

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Disaster Recovery Scenarios: What to Do When Things Go Wrong

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Chapter 9: Networking, Interfaces, VLANs, Routing, and Firewalling

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Network Interface Configuration: Single NIC, LAGG, and Bonding Modes

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

VLAN Configuration: Tagging, Trunking, and Isolated Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Advanced Networking: Bridges, Routing, and Multi-Homed Setups

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

DNS and DHCP: Integration with Existing Infrastructure

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Firewall Rules: Protecting Your NAS from Unauthorized Access

Chapter 10: File Sharing Services, SMB, NFS, iSCSI, and More

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

SMB/CIFS Configuration: Shares, Permissions, and Windows Integration

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

NFS Exporting: Versions, Security Models, and Unix Clients

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

iSCSI Targets: Block Storage for Virtual Machines and Servers

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

FTP and SFTP: Legacy File Transfer with Modern Security

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

WebDAV and Other Protocols: Specialized Use Cases

Chapter 11: Users, Groups, Permissions, and Access Control

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Local User and Group Management: Creating Accounts and Assigning Roles

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Active Directory Integration: Joining the Domain and Mapping Users

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

LDAP and Other Directory Services: Alternative Authentication Backends

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Filesystem Permissions: Unix Modes, SMB ACLs, and POSIX Compliance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

NFSv4 ACLs: Unified Access Control for Mixed Environments

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Chapter 12: Security Hardening, Monitoring, and Maintenance

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Security Baseline: Disabling Unused Services and Closing Attack Vectors

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

SSL/TLS Certificates: Self-Signed vs. CA-Signed for Web UI and Services

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Authentication Hardening: SSH Keys, Two-Factor Auth, and Password Policies

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Logging and Monitoring: System Logs, Alerts, and Integration with External Tools

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Routine Maintenance: Updates, Scrubs, Health Checks, and Documentation

Chapter 13: Advanced Topics, Automation, Migration, Performance Tuning, and Troubleshooting

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Migrating from TrueNAS CORE to FreeCORE: Strategies and Data Preservation

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Command-Line Administration: Essential zfs, zpool, and FreeCORE Commands

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Automation with Scripts and APIs: Configuring via CLI and REST API

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Performance Tuning: ARC, L2ARC, SLOG, and Kernel Parameters

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.

Systematic Troubleshooting: Diagnostics, Logs, and Recovery Procedures

Conclusion: Operating FreeCORE in Production, Lessons and Recommendations

References

This content is not available in the sample book. The book can be purchased on Leanpub at https://leanpub.com/freecore-thecompleteguide.