Build Your Own FreeBSD Server

About Me

Brief Introduction -- Nils Imhoff

- Works for as a DevOps-Engineer
- @work: mostly Ansible, Terraform, K8s, Linux
- @freetime: mostly BSD
- Reachable via nils@d4d1.de

My Repository

For the tutorial, I have prepared a repository:

```
git clone ssh://anonymous@got.d4d1.de:2225/bsdcan-tutorial-2026..git
```

and my slides

Target picture

Console-based private cloud and NAS under your control:

- `zroot` · `work` · `data`
- Bastille + VNET on shared bridges
- PF, file sharing, snapshots, updates

Schedule

Part	Topic
1	Basic server setup & ZFS
2	Networking & PF
3	Jails
4	Examples
5	Maintenance & backups

Appendices: SSO, Tailscale, OPNsense HAProxy + ACME, Cloud Services, Command Reference, Daemonless

Checklist (all in handouts)

Admin user · SSH · pkg · ZFS · PF · VNET · bridges ·
Bastille · jail pkg · file sharing · reverse proxy · Nextcloud · backup ·
snapshots · replication

Lab

`ssh tutorial@vmNN.d4d1.de` · `nda0` – `nda4` · handouts

Part 1 — Basics & ZFS

separate system / work / data

Pools

```
nda0       → zroot
nda1+2     → work  (mirror)  → bastille, bhyve
nda3+4     → data  (mirror)  → home, media, backups
```

At home: `data` is often raidz2 on many disks

work datasets

Dataset	Mount
`work/bastille`	`/usr/local/bastille`
`work/bhyve`	`/usr/local/bhyve`

Basics

- `pkg update && pkg upgrade`
- SSH: no root, no passwords (production)
- `bectl create pre-tutorial`

Part 2 — Networking & PF

SMART · Monit · VNET · bridges · firewall rules

S.M.A.R.T. + Monit

- `smartd` — scheduled self-tests, mail to root
- `monit` — load, ping, `zpool status`, filesystem fill

Network isolation

- **localnet0** — isolated jail network for services
- **publicnet0** — shared bridge to the LAN / WAN edge

`rc.conf.network` · `rc.conf.network-lab`

PF firewall

- `block-policy return`, `antispoof`, ICMP, SSH
- NAT: tables `<jails>` / `<vms>`
- `service pf reload`

Appendix: OPNsense HAProxy + ACME

Part 3 — Jails

`work/bastille` · VNET · `bridge0`

Why BastilleBSD?

- Thin layer over `jail.conf` + ZFS
- ZFS clones from the release snapshot
- Reusable templates for applications
- Coexists with BHYVE

Lab jails

Jail	IP
proxy	10.10.0.10
files	10.10.0.20
db	10.10.0.30

nullfs

- `data/bastille_data` → `/usr/local/bastille/data`
- per-jail `fstab`, match UID/GID
- lab shortcut: `bastille mount` → `/mnt/share`

Comparison: Classic jail vs iocage vs Bastille

Feature / Model	Classic `jail`	`iocage`	`Bastille`
Abstraction level	Low (manual)	Higher (property-driven)	Higher (template-focused)
ZFS integration	Manual	Strong (first-class)	Strong (clones, templates)
Ease of repeatable builds	Moderate (scripted)	Good (config properties)	Very good (Bastillefiles & templates)
Template / app focus	No	Some	Yes — templates for apps/jails
Target use	Fine-grained control	System jails with properties	App-focused jails, quick repros

Part 4 — Examples

Reverse proxy, file sharing, and application hosting

Reverse proxy / application hosting

- Nginx or HAProxy with SSL for public-facing apps
- Nextcloud as the example application

vm-bhyve

```
pkg install vm-bhyve bhyve-firmware
sysrc vm_dir="zfs:work/bhyve"
vm init
vm switch create -t manual -b bridge1 public
```

NFS (host → VM)

- Data on `data/`
- NFSv4 — not VirtIO-9p for DB workloads

Samba in a jail

`data` → nullfs → `files` → SMB

OCI apps (optional)

Daemonless — Podman or AppJail · `ghcr.io/daemonless/*`

Appendix: 09 Daemonless

More services

Nextcloud, Zabbix, NGINX …

Part 5 — Maintenance & backups

Snapshots · config backup · replication · updates

zfsnap2

- `data/home`, `data/media`, `work/bastille/jails`
- `zroot` → use `bectl` instead of snapshots

Back up config

`/etc`, `bastille`, `pf`, `monit`, `smartd`

Updates

- Host: `freebsd-update` + BE
- Jails: `bastille update` + `pkg upgrade`

Part 6 — Single sign-on

Take-home appendix. Don't try this in the room.

Why bother

- One identity = one off-boarding step
- MFA in one place (Passkeys at the IdP, not per-app)
- Family/team grows without a credentials sprawl

Cost: one more critical jail. Snapshot the dataset, plan a recovery.

Pick your path

Choose one path for identity in your deployment:

Path A — LDAP-only (recommended for mixed legacy + modern services)

```
389-ds (LDAP)
        ├─ Samba, SSH (LDAP)
        └─ apps that can speak LDAP
```

Path B — Keycloak-only (standalone, no LDAP required)

```
Keycloak (OIDC)
        └─ apps via OIDC (Nextcloud, Zabbix)
```

Path C — LDAP backend + OIDC layer (hybrid)

```
389-ds (LDAP) → Kanidm / Keycloak (OIDC gateway) → apps (OIDC)
```

App wiring

App	Path
Nextcloud	LDAP app + `user_oidc` (use both)
Samba	`passdb backend = ldapsam:ldaps://…`
Zabbix	LDAP + JIT provisioning, SAML opt.
SSH/host	`nss-pam-ldapd` → `getent passwd`
Legacy web	nginx + `oauth2-proxy` in `proxy` jail

Why not OIDC everywhere?

- Samba wants NT hashes
- NFSv4+Kerberos wants principals
- Legacy apps are forever
- Offline cache: LDAP yes, OIDC no

LDAP stays. OIDC is a layer, not a replacement.

Day-after homework

1. Stand up `idm` jail
2. Move yourself in (LDAP first, then OIDC)
3. Add LDAPS cert expiry to Zabbix
4. Replicate `data/idm` off-host
5. Write down a break-glass account on paper